BETA
SIMULATION SOLUTIONS

physics on screen

SPDRM
WORKFLOW MANAGER

Administrator's Guide

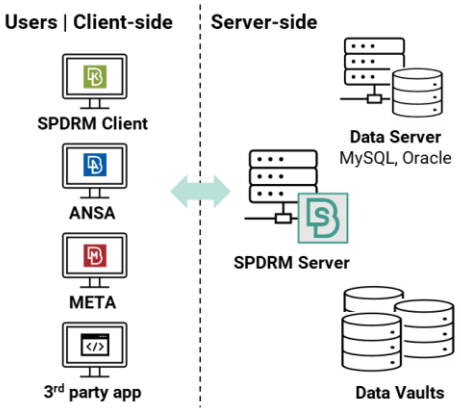
v1.8.0

Table of Contents

1. Introduction

1.1. What is SPDRM?

SPDRM is a software application for Simulation Process, Data and Resources Management that offers a unique CAE framework within which engineers and other simulation stakeholders can manage and archive their simulation data, share knowledge and collaborate.



Organization-wise, SPDRM is a database application with client/server architecture where the users (client-side) communicate with the SPDRM server over the network. As a distributed application, it can also be deployed globally, to cover the needs of geographically dispersed teams.

For Data Management, SPDRM comes with a CAE-ready yet customizable data model that covers all data management needs, from Parts and Sub-assemblies down to Simulation Runs, Reports and Optimization Studies. Its core functionality includes Version Control and Lifecycle Management, that enable comprehensive traceability from CAD Part to Simulation Runs and Reports. Data security is an integral part of SPDRM that offers role-based access control for all data and ensures that access is granted only to authenticated users.

For Process Management, SPDRM offers high-end Process Design capabilities that enable standardization and automation of processes through Process Templates. During Process Execution, information is tracked on how data were produced, by whom, through which applications and with which methodology. The built-in “HPC connector” assists the interfacing with HPC systems and enables effortless job submission and monitoring. Furthermore, the embedded utility “BETA Apps Launcher” enables the execution of time- and resource-consuming tasks on remote resources utilizing out-of-the-box load balancing and queuing capabilities.

The SPDRM Client is the front-end through which users browse and manage data and processes. Through the SPDRM Client, the users can search for data, review product structures, preview models in the embedded 3D viewer, send data to external applications, export data, review processed results, design and execute processes, get notifications on completed HPC jobs, and many more.

The first desktop client of SPDRM was a Java application that comes in the SPDRM installation. Since January 2022, a new, full featured client, KOMVOS, is introduced, that will gradually become the primary SPDRM client and will render the former Java application as obsolete for the end users. However still, the Java client will be the primary application for all administrative tasks.

SPDRM is an open system, and as such, it enables its seamless integration with other systems. For this purpose, SPDRM offers a REST API that enables direct access to Data, Process and Resources Management features of the system. Furthermore, it integrates the SPDRM Streamer, that enables real-time streaming of database changes to third party, analytics tools.

1.2. About this document

This document will attempt to provide to the administrator of the system all the needed information in order to set-up and configure SPDRM and its peripheral components in single-site and multi-site installations.

Information on the configuration of the new client, KOMVOS can be found in KOMVOS' User's Guide.

2. SPDRM architecture

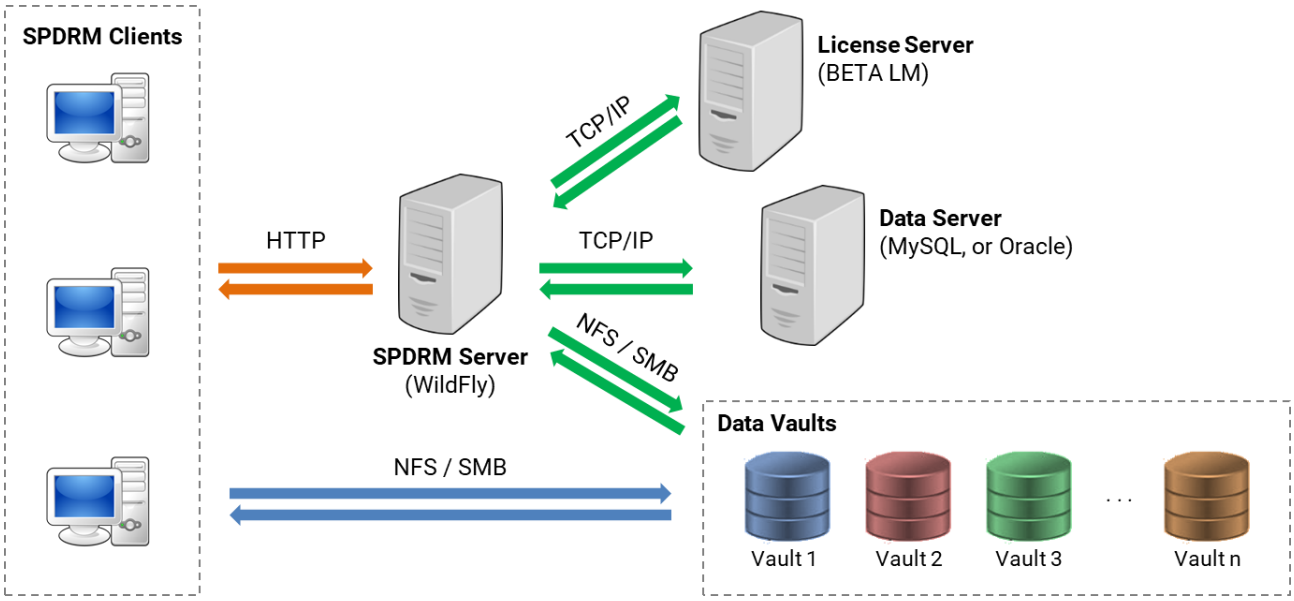
2.1. Single-site installation

The SPDRM software consists of two distinct programs:

- the SPDRM Server, and
- the SPDRM Client

The SPDRM architecture is based on a decentralized system, built on the concept of satellites. Being an Enterprise solution, the SPDRM Server and the Clients are installed on different workstations that communicate over a computer network. The data handled by SPDRM do not necessarily need to reside on the SPDRM Server.

The execution of processes (e.g. execution of scripts, launching of applications) takes place locally on the SPDRM Client's workstation and as a result, the SPDRM Server remains at a relatively low load, and is able to respond swiftly, keeping total traffic on minimum level. This is achieved by limiting the communication sessions between the SPDRM Clients and the Server to the absolutely necessary.



The **SPDRM Server** is the "back end" of SPDRM. The SPDRM Server responds to requests made by the SPDRM Clients by creating, querying and modifying database records.

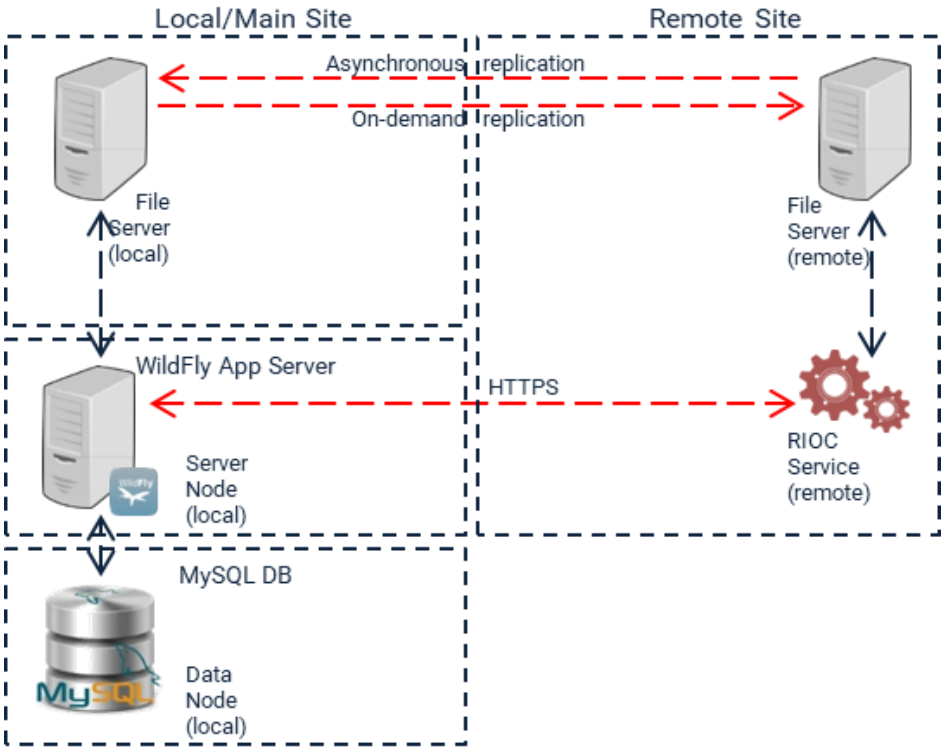
The **SPDRM Client** is the "front end" of SPDRM. The SPDRM Client offers the interface for the data, process and resources management.

Usually, the data reside on a separate **Data Server**, while the physical files are stored in one or more **Data Vaults**.

2.2. Multi-site installation

The **goal** of the SPDRM multi-site configuration is to enable efficient collaboration among globally dispersed CAE teams and suppliers.

The **main objective** of the multi-site solution is to provide local access to data that are stored in the file system (i.e. vault), on both SPDRM and ANSA clients that are located in different sites. In this way the time-consuming data transfers over the network will be minimized, or even eliminated.



The remote site vault is not synchronized real-time with the main one, but **on-demand files replication** is used instead, when downloading data from remote site.

File caching is supported on the remote vault. This means that when a file is requested by a remote client application (SPDRM Client or ANSA); it needs to be copied (using **SCP**) from the main vault to the remote vault only the first time. It will be available directly (locally) on all the subsequent requests of a remote client.

The server of the main site is responsible to perform the required remote file operations (copy, hard link) through **HTTPS**, on the remote site vault(s), with the aid of the **RIOC** service (see next section).

2.2.1. Pre-requisites

- A working SPDRM installation should exist on the local/main site (for more details about the SPDRM installation on the main site, please refer to the SPDRM Installation Guide).
- Select a Linux server machine that is located on the remote site to be used as the **remote file cache server** (**SSHhost**). The **SSHhost** is defined in *taxis.conf* (see next section).
- Create empty file structures of the vaults (firstDir, NodeExec, DM) on the remote file server to be used as **remote vaults** (for more details about how to configure vault, please refer to the SPDRM Installation Guide, chapter 4.5).
- Make a copy of the client installation directory from the main site to the file server on the remote site, to be used as the **remote client** (shared) installation directory.

- Create appropriate shares (using NFS or SMB protocol) to the **remote vaults** and the **remote client** paths so they will be accessible from the client machines that are located on the remote site.
- The **remote vaults** need to be mounted (using NFS or SMB protocol) on the client machines of the remote site.
- The **remote client** installation directory needs to be mounted on the client machines of the remote site.
- The **SPDRM Server** of the main site should be able to execute system commands (chown, chmod, mkdir, etc.) through **HTTPS**, on the **remote vaults**. This requires some additional configuration on the *sudoers* file on remote file server (see next section).
    - SSH should enable password-less login using ssh keys of the user that is used by the SPDRM server to perform SSH connection to the remote vaults (**SSHuser**). The **SSHuser** is defined in *taxis.conf* (see next section).
    - Additionally, an SSH login should be performed through the terminal in order to add the hostname of the machine that hosts the remote vaults to the list of known hosts that is stored in the local server machine.
- In case the exported file structure is activated, the path of **exported file structure** for the remote site need to be mounted on the client machines of the remote site, under the same absolute path as they defined on the client machines of the main site.

2.2.2. Required client configuration

The following additional key is required in the *taxisprops.xml* to enable the multi-site configuration (only for the client configuration file of the remote client installation):

```
<entry key="site">RemoteSite1</entry>
```

2.2.3. Required server configuration

The following additional keys are required in the *taxis.conf* to enable the multi-site configuration:

```
<entry key="sites">LocalSite,RemoteSite1</entry>
    (set names for local and remote sites)

<entry key="serverSite">LocalSite</entry>
    (define the site that hosts the server)

<entry key="RemoteSite1_cache">true</entry>
    (enable/disable file caching on RemoteSite1)

<entry key="RemoteSite1_SSHhost">172.24.45.215</entry>
    (hostname or IP address for file server of RemoteSite1)

<entry key="RemoteSite1_SSHport">22</entry>
    (port for SSH connection to file server of RemoteSite1)

<entry key="RemoteSite1_SSHuser">spdrm</entry>
    (user at RemoteSite1 to perform remote file operations)

<entry key="RemoteSite1_SSHkey">/home/spdrm/.ssh/id_rsa</entry>
    (path on local server that points to file that contains the ssh key in order to connect to file server of RemoteSite1 without password
    prompting)
```

```
<entry key="RemoteSite1_Hosts">hostname_1,hostname_prefix_2,hostname_prefix_3</entry>
    (define hostnames of client machines located at RemoteSite1)
    NOTE: The value "hostname_prefix_3" covers all client machines that their hostnames start with this particular prefix

<entry key="RemoteSite1_Addresses">10.125.34.3,10.168.56,10.234</entry>
    (define IP addresses and/or IP address ranges of client machines located at RemoteSite1)
    NOTE: The value "10.234" covers the IP address range from 10.234.0.0 to 10.234.255.255

<entry key="RemoteSite1_vault1_ssh">/share/spdrm/vaults/vault1/</entry>
    (path to the vault1 on file server of RemoteSite1)

<entry key="RemoteSite1_NodeExec_vault1_ssh">/share/spdrm/vaults/vault1/NodeExec/</entry>
    (path to NodeExec of vault1 on file server of RemoteSite1)

<entry key="RemoteSite1_export_vault1_ssh">/share/spdrm/vaults/vault1/DM/</entry>
    (path to exported file structure of vault1 on file server of RemoteSite1)

<entry key="RemoteSite1_NodeExec_L_vault1">/mnt/spdrm/vaults/vault1/NodeExec/</entry>
    (Linux path to be sent to ANSA Linux clients of RemoteSite1)

<entry key="RemoteSite1_NodeExec_W_vault1">S:\vaults\vault1\NodeExec\</entry>
    (Windows path to be sent to ANSA Windows clients of RemoteSite1)

<entry key="RemoteSite1_vault1_L_client_export">/mnt/spdrm/vaults/vault1/DM/</entry>
    (Linux path to be used for the synthesis of SPDRM links that targets to the exported file structure of RemoteSite1)

<entry key="RemoteSite1_vault1_W_client_export">S:\vaults\vault1\DM\</entry>
    (Windows path to be used for the synthesis of SPDRM links that targets to the exported file structure of RemoteSite1)
```

2.2.4. Required configuration on remote file server

The following lines need to be added in the /etc/*sudoers* file on remote file server to enable the multi-site configuration:

```
spdrm    ALL=(root)       NOPASSWD:/bin/chmod [0-7][0-7][0-7] /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/bin/chmod -R [0-7][0-7][0-7] /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/bin/chown * /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/bin/test -d /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/bin/test -e /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/bin/test -f /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/bin/mkdir /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/bin/cp /share/spdrm/vaults/vault1/* /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/bin/ln /share/spdrm/vaults/vault1/* /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/bin/ln -s /share/spdrm/vaults/vault1/*
/share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/bin/install -d -m [0-7][0-7][0-7] /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/bin/rm -rf /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/bin/scp -f /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/bin/scp -t /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/bin/ls -l /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/usr/bin/cd /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/bin/stat -c %d /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/bin/stat -c %i /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/bin/ls -p /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/usr/bin/wc -c /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/usr/bin/unzip -o /share/spdrm/vaults/vault1/*
spdrm    ALL=(root)       NOPASSWD:/bin/touch /share/spdrm/vaults/vault1/*
```

NOTE: The name of the technical account (spdrm) and the vault path that is used in the above example needs to be replaced accordingly.

2.2.5. Setup of RIOC service (remote HTTPS server)

This section provides instructions for the installation of SPDRM single-server multi-site solution (Vault Replication) with the use of **RIOC** (Remote Input-Output Commands) external service (i.e. remote HTTPS server) to handle file I/O commands, related to remote client requests.

2.2.5.1. Topology

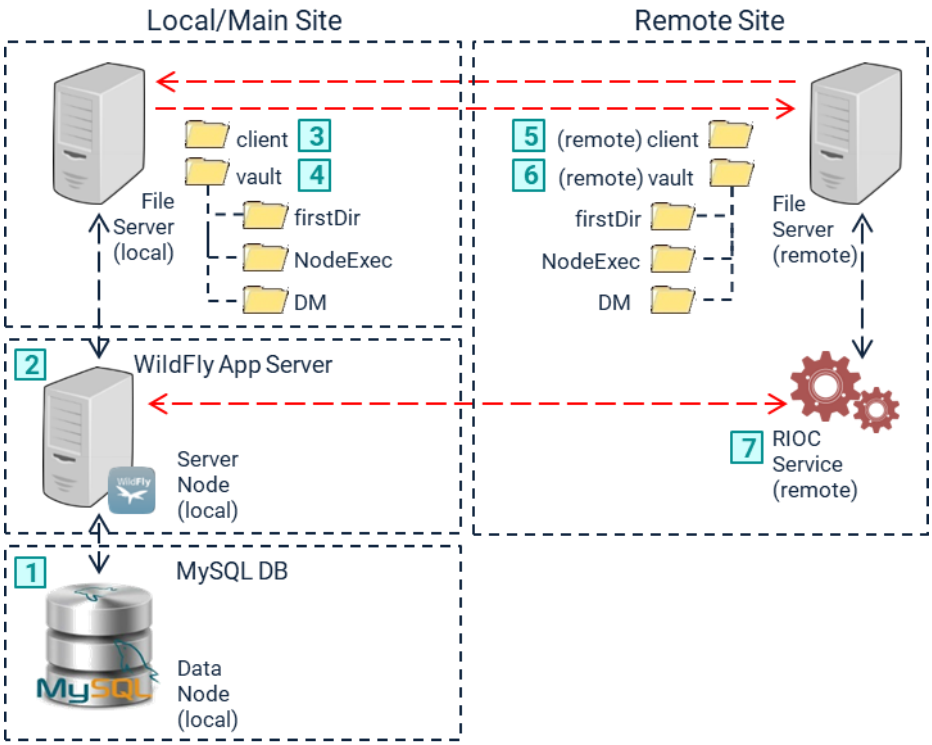
In the following instructions it is assumed that the desired SPDRM multi-site architecture consists of 2 different sites:

The local/main site that hosts the following system components:

1. SPDRM database server (MySQL)
2. SPDRM application server (WildFly)
3. SPDRM client installation directory (shared)
4. SPDRM vaults (shared)

The remote site that hosts the following system components:

5. SPDRM remote client installation directory (shared)
6. SPDRM remote vaults (shared)
7. SPDRM RIOC service (i.e. remote HTTPS server)



The SSL keys that are required for the setup of the secure communication between the SPDRM Server and the RIOC service are included in the package.

NOTE: Optionally, these SSL keys could be replaced by keys that can be provided by customer's IT department.

The following procedure is followed in order to produce the provided **SSL keys** for the RIOC service:

- export keystore:
  ```
  keytool -genkeypair -keystore spdrm.jks -storepass spdrm123 -keypass spdrm123 -keyalg RSA -validity 3650 -alias spdrm-certificate -dname "cn=spdrm-security,o=BetaCAE,c=GR"
  ```
- export public certificate:
  ```
  keytool -export -alias spdrm-certificate -keystore spdrm.jks -rfc -file spdrm-cert.crt
  ```
- create client truststore (named spdrm-client.jks) and store the public certificate there:
  ```
  keytool -import -alias spdrm-certificate -file spdrm-cert.crt -keystore spdrm-client.jks
  ```

The spdrm.jks is the SSL key for the RIOC service, and should be placed in:
`<SPDRM_REMOTE_CLIENT_DIR>/RIOC/RIOCSSL/`

The spdrm-client.jks is the SSL key for the SPDRM application server, and should be placed in:
`<SPDRM_SERVER_DIR>/server/wildfly/standalone/configuration/spdrm/`

2.2.5.2. Configuration on the local/main site

- WildFly Application Server Configuration (standalone-full.xml)

  The SSL related information should be declared in the *standalone-full.xml*.

  To do so, the following properties should be added in the system-properties section:

  ```
  <system-properties>

  ..

  <property name="javax.net.ssl.trustStore" value="/path/to/spdrm-client.jks"/>

  <property name="javax.net.ssl.trustStorePassword" value="spdrm123"/>

  </system-properties>
  ```

- SPDRM Server Configuration (taxis.conf)

  The following entries should be added in the taxis.conf to define:

  - the IP address of the remote server machine that hosts the HTTPS server (RIOC service)
  - the port used for the communication between the SPDRM server and the RIOC service

  ```
  <entry key="file_io_microservice_address_RemoteSite">RIOC_IP_ADDRESS</entry>

  <entry key="file_io_microservice_port_RemoteSite">3457</entry>
  ```

NOTE: The RemoteSite string should be replaced by the name of the remote site.

2.2.5.3. Configuration on the remote site

RIOC Service Installation

The provided package RIOC.zip should be extracted to the SPDRM remote client installation directory.

The following files and paths should exist during the initial setup (included in RIOC.zip):

- `<SPDRM_REMOTE_CLIENT_DIR>/RIOC/`
- `<SPDRM_REMOTE_CLIENT_DIR>/RIOC/log/`

- <SPDRM_REMOTE_CLIENT_DIR>/RIOC/RIOCSSL/
- <SPDRM_REMOTE_CLIENT_DIR>/RIOC/RIOCSSL/spdrm.jks
- <SPDRM_REMOTE_CLIENT_DIR>/RIOC/RIOCSSL/spdrm-client.jks
- <SPDRM_REMOTE_CLIENT_DIR>/RIOC/temp/
- <SPDRM_REMOTE_CLIENT_DIR>/RIOC/application.properties
- <SPDRM_REMOTE_CLIENT_DIR>/RIOC/RIOC.jar
- <SPDRM_REMOTE_CLIENT_DIR>/RIOC/stopRIOC.sh
- <SPDRM_REMOTE_CLIENT_DIR>/RIOC/startRIOC.sh

RIOC Service Configuration (application.properties)

The configuration file of the RIOC service is called application.properties, and is located in the same path with RIOC.jar.

An example of the application.properties file is given bellow, along with a short description of the property keys:

```
# Define a custom port to be used by the RIOC service
server.port=3457

# Control whether to accept requests only through HTTPS, or not
security.require-ssl=true
logging.level.com.betacae.service=DEBUG
logging.file=./log/RIOC.log
spring.output.ansi.enabled=always

# The full path to temp folder which will be used by RIOC service
temp.path=./temp/

# Define whether the temp file created by the RIOC service is kept, or not. It should be enabled only for debugging purpose.
keep.temp.file=false

# The IP address for the client of the RIOC service (i.e. SPDRM Server IP address)
clientIP=SPDRM_SERVER_IP_ADDRESS

# The format used for the keystore
server.ssl.key-store-type=JKS

# The path to the keystore containing the certificate
server.ssl.key-store=./RIOCSSL/spdrm.jks

# The password used to generate the certificate
server.ssl.key-store-password=spdrm123

# The alias mapped to the certificate
server.ssl.key-alias=spdrm-certificate
```

2.2.5.4. Start RIOC Service

IMPORTANT: The RIOC service must be started with root privileges.
If this is not possible due to security restriction, then it could be started by the same user ID that is currently used by the SPDRM Server to connect via SSH to the remote site (i.e. RemoteSite_SSHuser in *taxis.conf*).

The RIOC service (remote HTTPS server) can be launched using the command:

```
<SPDRM_REMOTE_CLIENT_DIR>/RIOC/startRIOC.sh
```

An initial log file that contains the initialization logs is created during the start-up of the remote HTTPS server, in the path:

```
<SPDRM_REMOTE_CLIENT_DIR>/RIOC/startup-log.txt
```

All other logs will be written out in the following path:

```
<SPDRM_REMOTE_CLIENT_DIR>/RIOC/log/RIOC.log
```

2.2.5.5. Stop RIOC Service

The RIOC service (remote HTTPS server) can be stopped using the command:

```
<SPDRM_REMOTE_CLIENT_DIR>/RIOC/stopRIOC.sh
```

3. System configuration

3.1. MySQL database server configuration and optimization

To optimize the performance of the MySQL server for SPDRM usage on production environments, you should modify the **my.cnf** file by adding the following properties:

```
innodb_buffer_pool_size = 4G*
```

```
innodb_log_file_size = 1G*

innodb_log_buffer_size = 100M
```

```
innodb_flush_log_at_trx_commit = 1

innodb_lock_wait_timeout = 500
```

* The values of the properties that are highlighted using red fonts should be selected according to the following rules:
  - `innodb_buffer_pool_size`: the value should be greater than 2G, and up to 50% of RAM on the machine that hosts the MySQL server.
  - `innodb_log_file_size`: the value should be equal to 25% of the innodb_buffer_pool_size.

To define MySQL startup options on the option file please refer to:
https://dev.mysql.com/doc/refman/5.6/en/option-files.html

After saving the updated **my.cnf** file, you should perform the following steps:

- enter the MySQL command line from a terminal window (i.e. mysql -u root -p)
- run the following query:
  - `mysql> SET GLOBAL innodb_fast_shutdown = 0;`
- Stop the SPDRM Server and the MySQL
- Backup and then delete the following files from the source location:

  (these files are located either in: /var/lib/mysql/ , or in the data directory of MySQL, which is defined in the startup script of mysql (i.e. --datadir=<data_directory_of_MySQL>))

  - ib_logfile0
  - ib_logfile1
- Start the MySQL and SPDRM Server

Finally, please enter the MySQL command line from a terminal window (i.e. **mysql -u root**) and run the following query (for validation purpose):

- `mysql> show variables like '%innodb%';`

The above query returns (among others) the new values (in bytes) for the properties that were changed in the previous step.

3.2. WildFly application server

3.2.1. Access Admin Console

The SPDRM Server application is deployed on the WildFly application server.

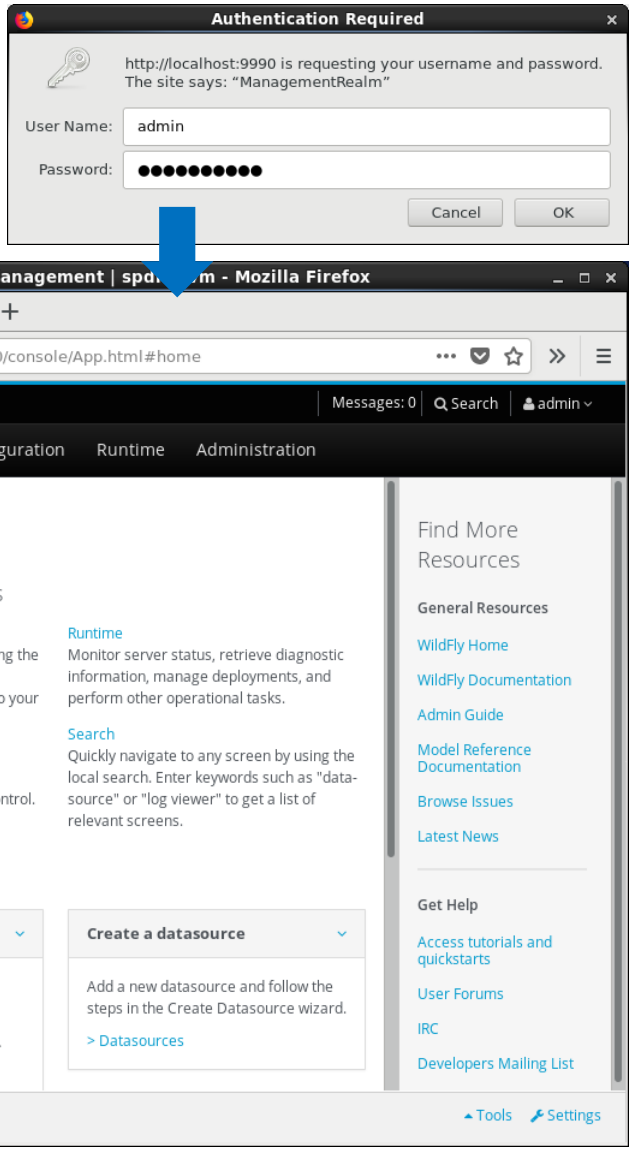
The WildFly 8.2.1 application server is automatically installed and configured during the installation of SPDRM.

To access the WildFly admin console:

1. Open a Web Browser and go to

   `http://<SPDRM_SERVER_HOST>:9990`

2. Login using the fo llowing credentials:
   - User Name: *admin*
   - Password: *adminadmin*



3.2.2. Configuration

3.2.2.1. Change location of server logs

The WildFly application server produces logs on two different files:

- **server.log** (contains logs related to WildFly application server)
- **spdrm.log** (contains logs related to SPDRM application)

By default the log files are located in:

- `<SPDRM_SERVER_DIR>/wildfly/standalone/log/server.log`
- `<SPDRM_SERVER_DIR>/wildfly/standalone/log/spdrm/spdrm.log`

To modify the default location of these server logs you should:

- edit the **logging.properties** file that is located in:

  `<SPDRM_SERVER_DIR>/wildfly/standalone/configuration/`

  - modify the key: handler.FILE.fileName

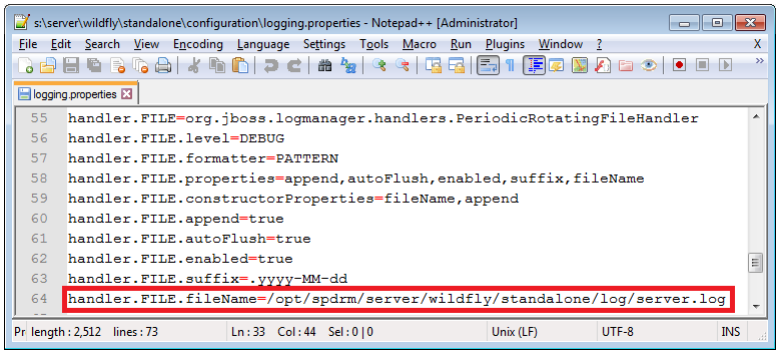
- edit the **standalone.conf** file that is located in:

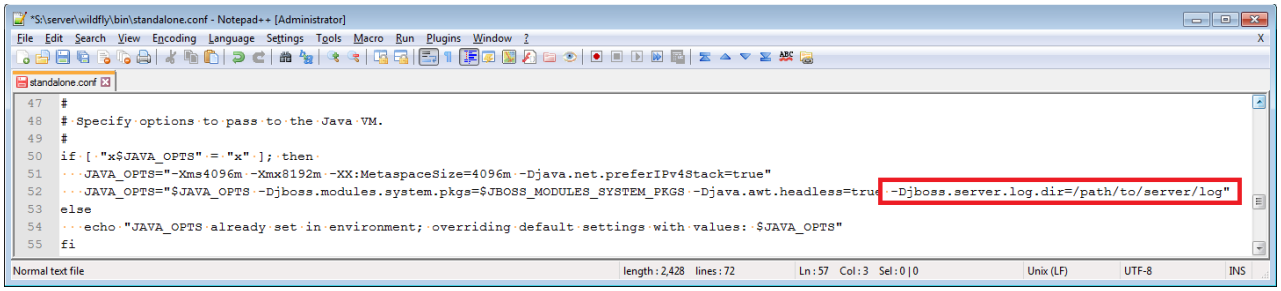
  `<SPDRM_SERVER_DIR>/wildfly/bin/`

  - append the argument:

    `-Djboss.server.log.dir=<NEW_LOG_DIR_PATH>` in the `JAVA_OPTS` variable.

NOTE: The changes will take effect after the restart of the server.





3.2.2.2. Server log rotation policy

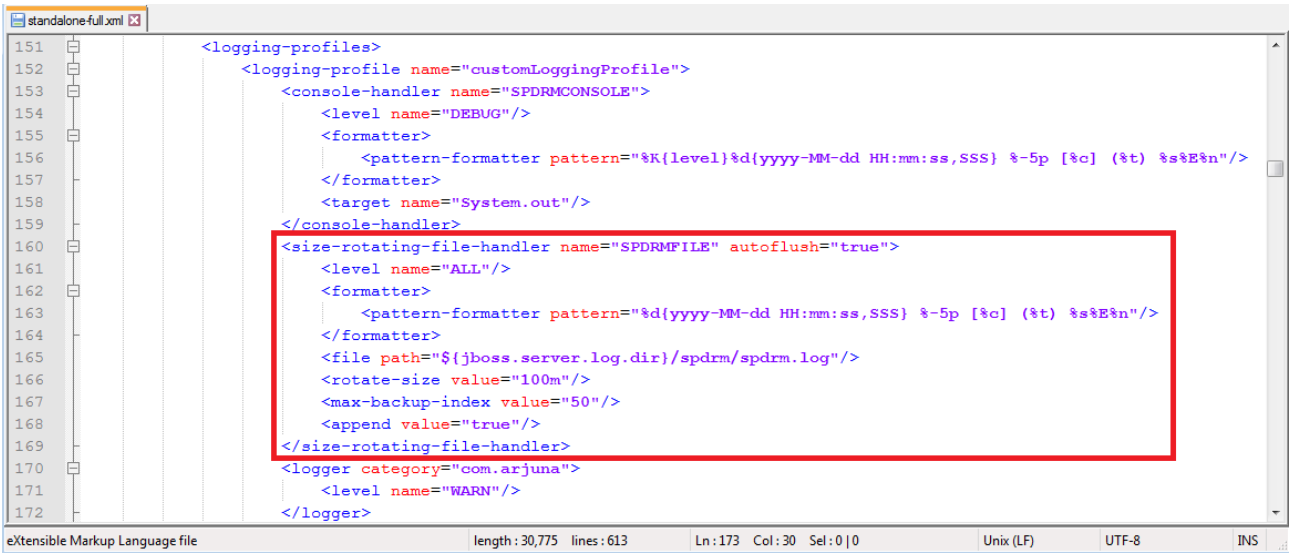
The WildFly application server supports two different rotating methods for the log files:

- periodic-rotating
- size-rotating

By default, the periodic-rotating is used. This creates new log files (i.e. server.log and spdrm.log) every day, archiving the previous one under the name server.log.yyyy-MM-dd and spdrm.log.yyyy-MM-dd.

To enable the size-rotating method you should edit the *standalone-full.xml* and replace the periodic-rotating-file-handler section (under the logging-profiles section) with the following section:

```
<size-rotating-file-handler name="SPDRMFILE" autoflush="true">
      <level name="ALL"/>
      <formatter>
            <pattern-formatter pattern="%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c]
            (%t) %s%E%n"/>
      </formatter>
      <file path="${jboss.server.log.dir}/spdrm/spdrm.log"/>
      <rotate-size value="100m"/>
      <max-backup-index value="50"/>
      <append value="true"/>
</size-rotating-file-handler>
```



NOTES:

- The rotate-size value that is used in the above example is 100 MB, but it can be changed according to your need.
- The max-backup-index that is used in the above example is 50 archived log files, but it can be customized.
- The above changes will take effect after the restart of the application server.

3.2.3. Email Service Configuration

The SPDRM offers several features that require the configuration of the email service in Wildfly application server:

To enable the email service the following steps should be followed:

1. Edit the configuration file of the Wildfly application server, located in:

   `[SPDRM_SERVER_DIR]/wildfly/standalone/configuration/standalone-full.xml`

2. Replace the following parameters with the actual values:
   - **[EMAIL]:** A technical mail account that will be used by the SPDRM server for the authentication to the outgoing mail server (SMTP) (e.g. *spdrm@domain.com*)
   - **[PASSWORD]:** The password that is required for the authentication of the above user account to the mail server
   - **[HOST]:** The host name of the mail server (e.g. *mail.domain.com*)
   - **[PORT]:** The port of the mail server (e.g. *25*)
3. Uncommon the respective sections (lines 559-563, 712-716)
4. Save the modifications on the file
5. Restart the application server to apply the new settings

```
555     <subsystem xmlns="urn:jboss:domain:mail:3.0">
556         <mail-session name="default" jndi-name="java:jboss/mail/Default">
557             <smtp-server outbound-socket-binding-ref="mail-smtp"/>
558         </mail-session>
559         <!--
560         <mail-session name="java:jboss/mail/Taxis" jndi-name="java:jboss/mail/Taxis">
561             <smtp-server outbound-socket-binding-ref="mail-smtp-spdrm" ssl="false" username="[EMAIL]" password="[PASSWORD]"/>
562         </mail-session>
563         -->
564     </subsystem>
```

```
712     <!--
713     <outbound-socket-binding name="mail-smtp-spdrm">
714         <remote-destination host="[HOST]" port="[PORT]"/>
715     </outbound-socket-binding>
716     -->
```

NOTE: In order to enable the **StartTLS** protocol, the **tls="true"** argument must be added after the password argument in the **outbound-socket-binding-ref** argument.

3.3. Email Notification using SPDRM Hyperlinks

SPDRM support its own hyperlinks on email notifications. These hyperlinks can trigger SPDRM operations (e.g. view process node, execute process node, search data).

Examples of SPDRM Hyperlinks:

- View node
  spdrm:///process?action=viewNode&nodeId=11106
- Execute node
  spdrm:///process?action=executeNode&nodeId=11106
- Search data
  spdrm:///dm?action=showDMItems&selectionType=ENTITIES&dmItemIds=3214,3216,3301

The following optional keys could be used in order to customize this functionality:

- To change the default port (i.e. 1500) on which the client will be listening for commands, the following entry should be added in the client's configuration file (*taxisprops.xml*), e.g.

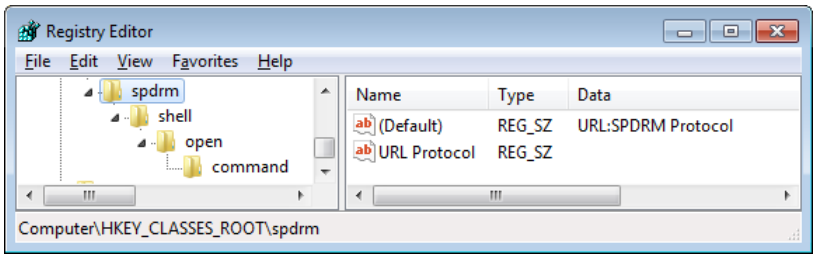
  `<entry key="command.port">1234</entry>`

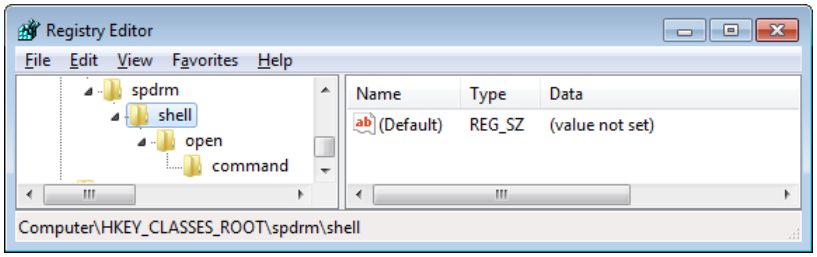
- To enable the users and the underlining OS to associate the hyperlinks with different SPDRM environments (e.g. dev, prod), a suffix could be used in order to synthesize the SPDRM hyperlinks

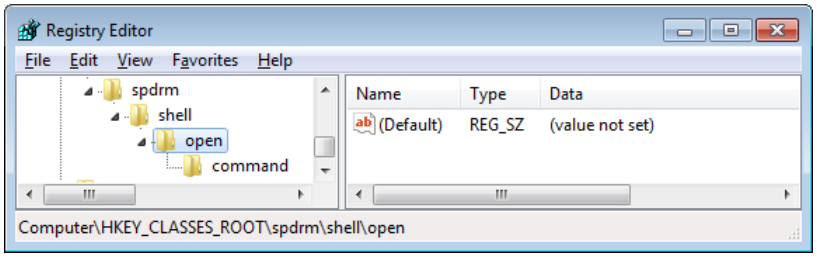
(e.g. spdrm-prod://... , spdrm-dev://...). To customize the links, the following entry should be added in the server's configuration file (*taxis.conf*), e.g.

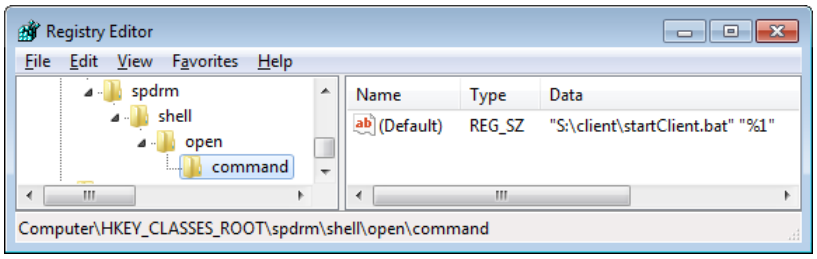
```
<entry key="link_suffix">prod</entry>
```

To enable this functionality on Windows client machines, the registry keys shown in the images below, should be added.









3.4. Data share via Email using SPDRM Hyperlinks

SPDRM offers the option to share information about data objects (e.g. Subsystems) with other users, via email. This feature is enabled by the context menu option **Send email**, in the Search lists. It automatically invokes the embedded email composer of SPDRM, with pre-filled information about the selected data objects, in the form of **internal link** (a.k.a SPDRM hyperlink) and **external links** (in Windows and Linux format).

If the email recipient has access to the SPDRM Client, then the **internal link** (SPDRM hyperlink) can be used, that will open the SPDRM Client and list the particular data objects in a Search window, e.g.:

spdrm:///dm?action=showDMItems&selectionType=ENTITIES&dmItemIds=3214,3216,3301

Alternatively, the email recipient (either SPDRM or non-SPDRM user) can use the **external links** (for Windows or Linux OS), that will redirect to the file paths into the external data structure, e.g.:

S:\vaults\vault1\DM\Entities\Subsystems\EU\VENZA\A00\CRASH\FE\108_HOOD_A00_EU_CRASH_FE_0.key

S:\vaults\vault1\DM\Entities\Subsystems\EU\VENZA\A00\CRASH\FE\105_DOOR_RR_A00_EU_CRASH_FE_0.key

S:\vaults\vault1\DM\Entities\Subsystems\EU\VENZA\A00\CRASH\FE\104_DOOR_FR_A00_EU_CRASH_FE_0.key

Note that the **external links** are only available to SPDRM environments where the external file structure is activated.

In order for the SPDRM server to compose the proper absolute path of the **external links**, the following entries should be declared for each Vault, OS and Site in the SPDRM Server's configuration file (**taxis.conf**), e.g.:

```
<entry key="vault1_L_client_export">/mnt/spdrm/vaults/vault1/DM/</entry>
<entry key="vault1_W_client_export">S:\vaults\vault1\DM\</entry>
<entry key="RemoteSite1_vault1_L_client_export">/mnt/spdrm/vaults/vault1/DM/</entry>
<entry key="RemoteSite1_vault1_W_client_export">S:\vaults\vault1\DM\</entry>
```

NOTE: The letter **"L"** in the above keys stands for Linux OS, whereas the letter **"W"** stands for Windows OS.

3.5. SPDRM API

3.5.1. Web Services

3.5.1.1. Basic

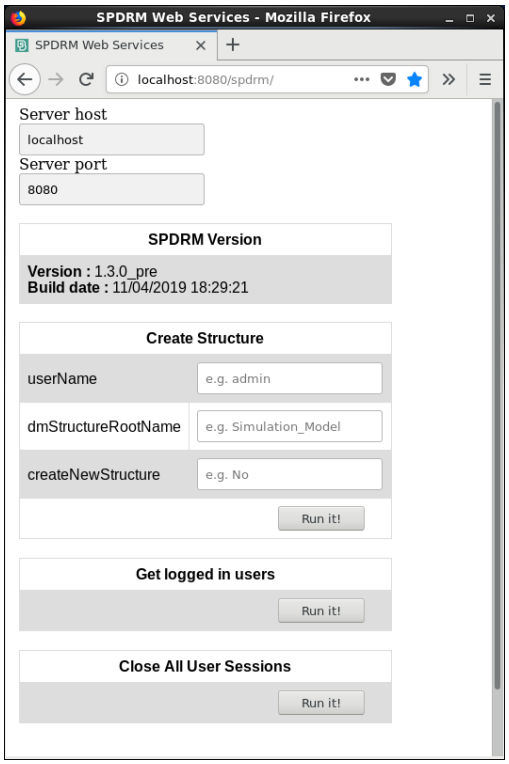
The SPDRM Server exposes a set of basic functionality through Web Services. These Web Services can be used to read, write and modify data in the SPDRM Database, through the SPDRM Server.

To access the most commonly used SPDRM Web Services open a Web Browser and go to:

```
http://<SPDRM_SERVER_HOST>:8080/spdrm
```

This page enables access to the following SPDRM web services:

- **SPDRM Version**: Displays the SPDRM server version and build date.
- **Create Structure**: Updates the SPDRM data model / data views.
- **Get logged in users**: Returns the list of logged-in users.
- **Close all user sessions**: Closes remotely the SPDRM Client sessions of all logged-in users.

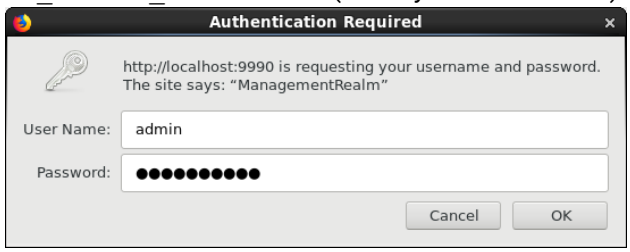


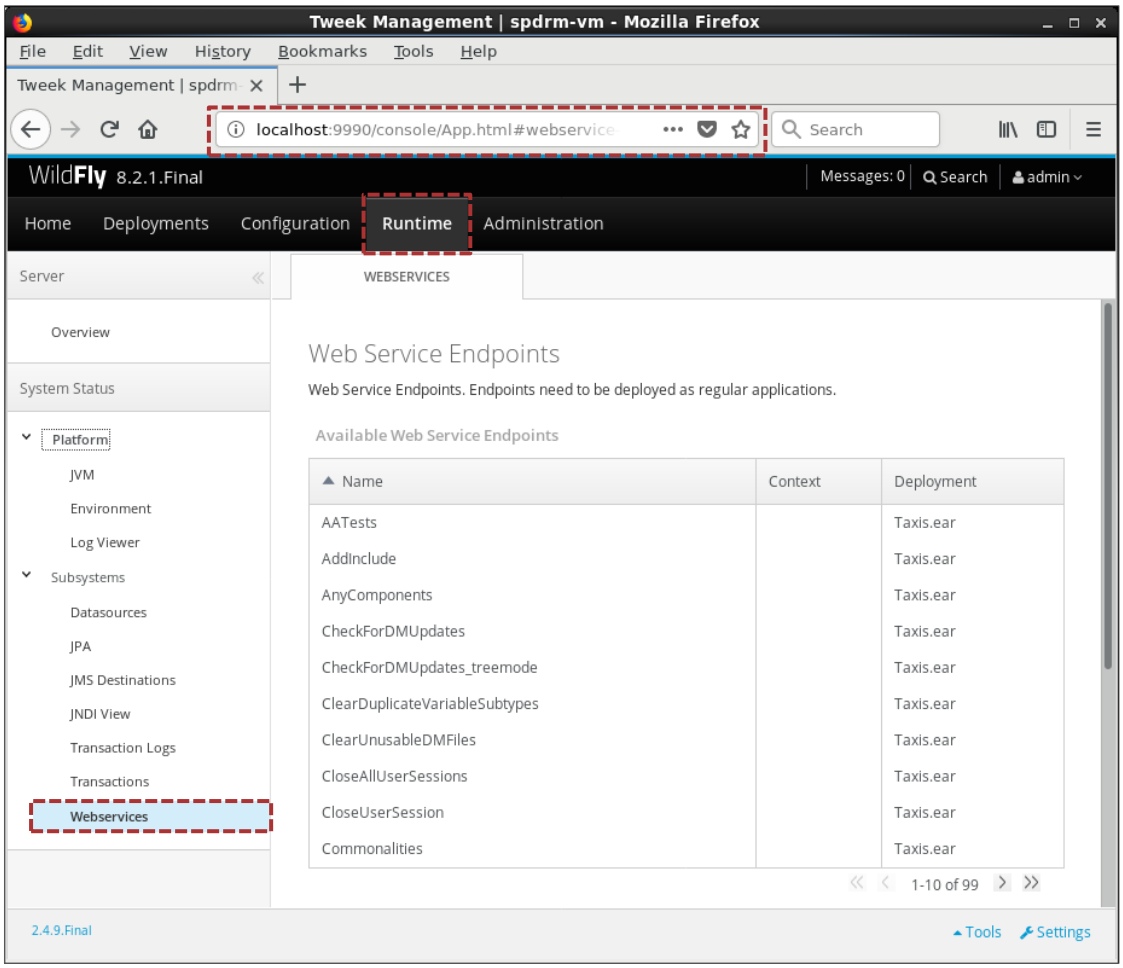
3.5.1.2. Advanced

The SPDRM Server exposes a set of basic functionality through Web Services. These Web Services can be used to read, write and modify data in the SPDRM Database, through the SPDRM Server.

To access the full list of SPDRM Web Services:

1. Open a Web Browser and go to `http://<SPDRM_SERVER_HOST>:9990` (WildFly Admin console).
2. Login using the following credentials:
   - User Name: *admin*
   - Password: *adminadmin*
3. Switch to the **Runtime** tab
4. Select the item **Webservices** from the tree on the left
5. The list of available Web Services is displayed on the right

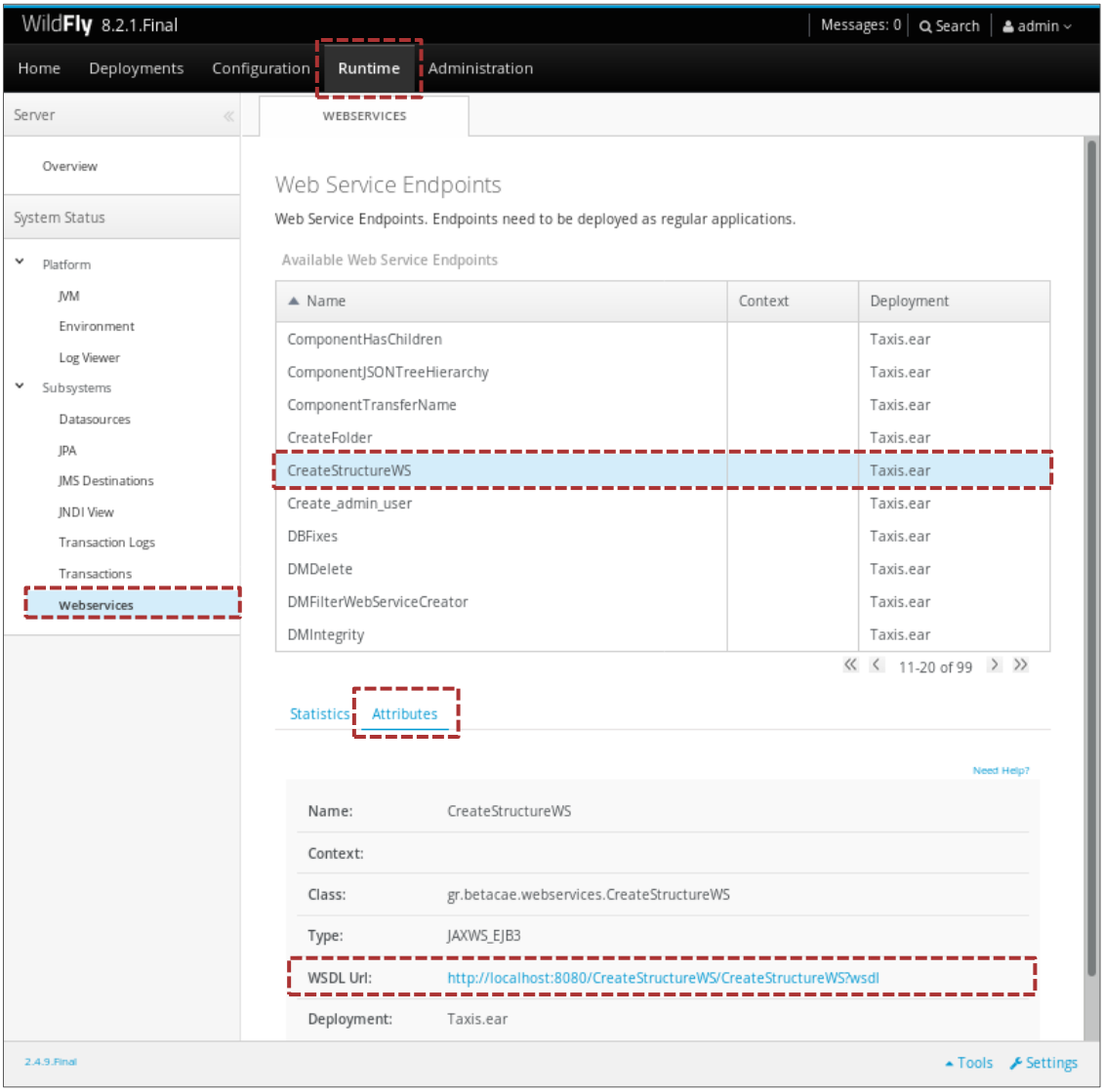




3.5.2. Execute web services

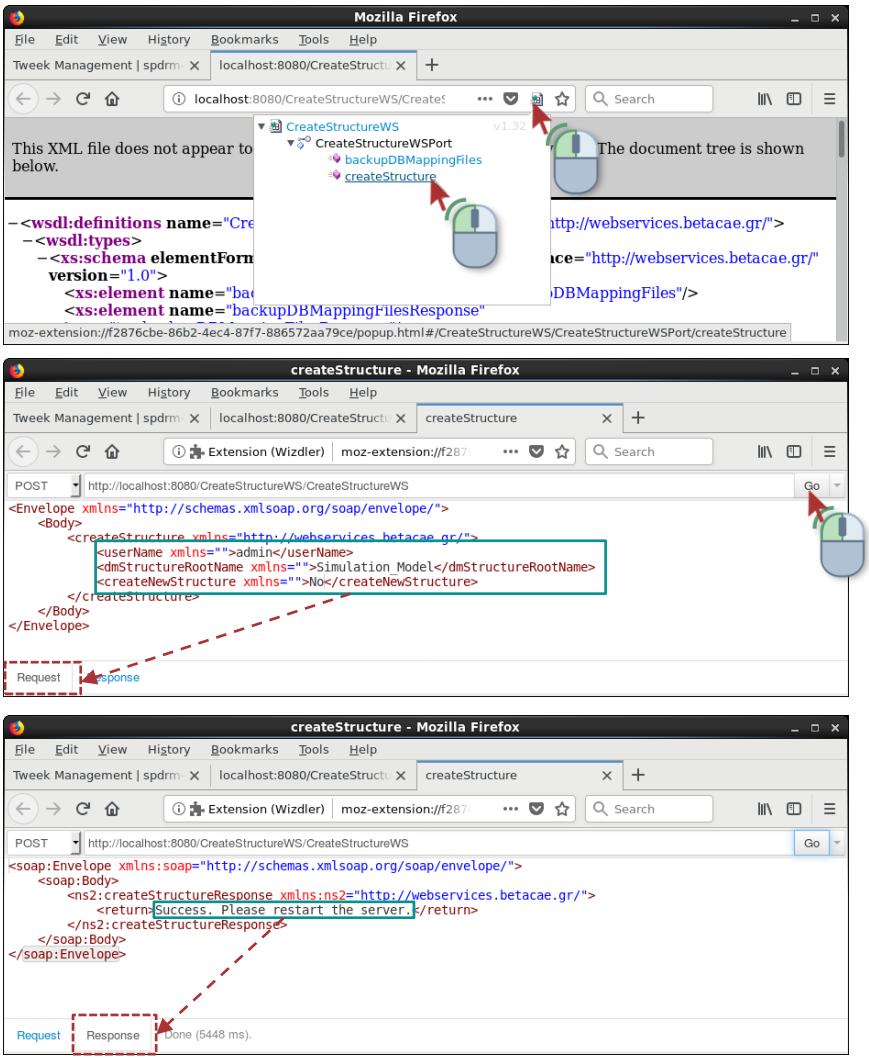
To execute a SPDRM Web Service:

1. Login to the WildFly Admin console (`http://<SPDRM_SERVER_HOST>:9990`)
   - User Name: *admin*
   - Password: *adminadmin*
2. Switch to the **Runtime** tab
3. Select the item **Webservices** from the tree on the left
4. Select a Web Service from the list (e.g. CreateStructureWS)
5. Switch to the **Attributes** tab
6. Click on the **WSDL Url** link



NOTE: In order to be able to run Web Services on WildFly Application Server, a SOAP add-on (e.g. Wizdler) should be first installed on the Web browser.

7. Click on the icon that appears next to the URL.
8. Select the **createStructure** Web Service method.
9. In the new browser tab that will pop-up, edit the XML request to set the required input arguments.
10. Press the **Go** button to execute the Web Service method.
11. The response of the Web Service appears in the **Response** tab.



3.6. SPDRM Server Configuration

3.6.1. Required settings on taxis.conf

The configuration settings of the SPDRM Server are included in the ***taxis.conf*** file that is located in: `[SPDRM_SERVER_DIR]/wildfly/standalone/configuration/spdrm/`.

IMPORTANT: The directory paths that are used in this file should always end with a trailing slash.

Key	Type	Default value	Description
vaults	string		Comma separated values that define the name of the SPDRM vaults.
defaultVault	string		The name of the vault that will be used as default storage location of files, in case the vault information is not set explicitly by the user/application. It makes sense only on SPDRM installations that use multiple vaults.
<vaultName>	string		The full path to the <vaultName> vault, as it is accessible by the server machine. **NOTE:** In case that multiple vaults are used, then this key should be defined for each vault.
export_<vaultName>	string		The full path to the exported data structure of the <vaultName> vault, as it is accessible by the server machine. The exported data structure is a read-only precise representation of the SPDRM data structure. It can be optionally created and maintained by the SPDRM Server, on a designated location on the file system. By default the mechanism that creates and maintain the exported data structure is disabled. It is activated and configured through: *Tools > DM structure export setup*. **NOTE:** In case that multiple vaults are used, then this key should be defined for each vault.
W_<vaultName>	string		The full path to the <vaultName> vault, as it is accessible by ANSA/META Windows clients. It is used for the communication of data between the SPDRM Server and a Windows-based ANSA/META clients. This path should be shared to client machines granting read and execute permissions. It should be mounted on client machines using a network drive letter. **NOTE:** In case that multiple vaults are used, then this key should be defined for each vault.
L_<vaultName>	string		The full path to the <vaultName> vault, as it is accessible by ANSA/META Linux clients. It is used for the communication of data between the SPDRM Server and a Linux-based ANSA/META clients. This path should be shared to client machines granting read and execute permissions. It should be mounted on client machines using a network drive letter. **NOTE:** In case that multiple vaults are used, then this key should be defined for each vault.
<vaultName>_directorySize	integer	51200	The maximum number of files that can be stored in the “firstDir” directory of the <vaultName> vault. **NOTE:** In case that multiple vaults are used, then this key should be defined for each vault.

Key	Type	Default value	Description
nodeExecPath	string	NodeExec/	Defined the path where the SPDRM Server will create the folders (temporary working directories) for every operation that requires file I/O (e.g. node execution). This path is defined relatively to the vault path.
securityEnabled	boolean	true	Defines if the files/folders that are created by the SPDRM Server are owned by the user who triggered the file I/O into the vault, or if files/folders are accessible by everyone. When this key is **true** the "chown" operation applies on the files during importing actions in SPDRM. When this key is **false** the "chown" operation is not applied on the files during importing actions in SPDRM. The "copy" mechanism is used, instead. Please refer to the Appendix B of the SPDRM Installation Guide for more details about the SPDRM security policy
commonSML	string	u+rwx,go=rx	Defines the common symbolic mode list for the common files (i.e. files accessible by any user). Applies to the files that are exported from SPDRM (into the "NodeExec" directory). The permissions are specified with a symbolic notation, as a combination of references and modes, similarly to the syntax of the chmod command (e.g. u+rwx,go=rx adds read, write and execute permissions to the user (owner) and assign read and execute permissions for his/her group and others).
privateSML	string	u+rwx,go=	Defines the private symbolic mode list for the private files (i.e. files accessible only by their owner). Applies to the files that are imported to SPDRM (into the "firstDir" directory). **NOTE:** This affects only the SPDRM Server configuration with securityEnabled = true.
win_user_group	string	SPDRM_users	This key is required only for SPDRM Server that is installed on Windows OS. Defines the name of the Windows group that includes all the users that are going to use SPDRM. This is required in order for the SPDRM Server to be able to set the appropriate file permissions during Client-Server I/O.

3.6.2. Optional settings on taxis.conf

NOTE: The keys in the following tables that refer to <siteName> are required to be defined only for the remote sites of an SPDRM multi-site installation.

Key	Type	Accepted values	Default value
<siteName>_Addresses	string		
The IP addresses (comma separated values) of the client machines located at the <siteName> site. These values are used by the SPDRM Server to locate the site of the HTTP client application (e.g. ANSA, META), when receiving a HTTP request. **NOTE:** IP address prefixes are also supported by this key, e.g. the value "*10.234*" covers the IP address range from *10.234.0.0* to *10.234.255.255*.			
<siteName>_cache	boolean	true, false	
It controls the file caching on the <siteName> site.			
<siteName>_Hosts	string		
The hostnames (comma separated values) of the client machines located at the <siteName> site . These values are used by the SPDRM Server to locate the site of the HTTP client application (e.g. ANSA, META), when receiving a HTTP request. **NOTE:** Hostname prefixes are also supported by this key, e.g. the value "hostname_prefix" covers all client machines that their hostnames start with this particular prefix.			
<siteName>_SSHhost	string		
The hostname, or IP address that is used by the SPDRM Server for the SSH connection to the file server of the <siteName> site.			
<siteName>_SSHkey	string		
The path (on the SPDRM Server machine) that points to the file that contains the SSH key in order to connect to the file server of the <siteName> site without password prompting.			
<siteName>_SSHport	integer		22
The port that is used by the SPDRM Server for the SSH connection to the file server of the <siteName> site.			
<siteName>_SSHuser	string		
The SSH user to perform the file operations on the file server of the <siteName> site, on behalf of the SPDRM Server.			
<siteName>_<vaultName>_ssh	string		
The path to the <vaultName> vault on file server of the <siteName> site.			
<siteName>_NodeExec_<vaultName>_ssh	string		
The path to the NodeExec directory of the <vaultName> vault on file server of the <siteName> site. This is used by the SPDRM Server for remote file I/O operations in the <vaultName> vault.			
<siteName>_export_<vaultName>_ssh	string		

The path to the exported data structure of the <vaultName> vault on file server of the <siteName> site.			
Key	**Type**	**Accepted values**	**Default value**
<siteName>_NodeExec_L_<vaultName>	string		
The Linux mounted path to be used by ANSA/META Linux clients of the <siteName> site, for all file I/O operations in the <vaultName> vault.			
<siteName>_NodeExec_W_<vaultName>	string		
The Windows mounted path to be used by ANSA/META Windows clients of the <siteName> site, for all file I/O operations in the <vaultName> vault.			
<siteName>_<vaultName>_L_client_export	string		
The Linux path to be used for the synthesis of SPDRM hyperlinks that targets to the exported data structure of the <vaultName> vault on file server of the <siteName> site.			
<siteName>_<vaultName>_W_client_export	string		
The Windows path to be used for the synthesis of SPDRM hyperlinks that targets to the exported data structure of the <vaultName> vault on file server of the <siteName> site.			
<vaultName>ArchiveDeleted	string		
Defines the path where SPDRM Server will archive files that have been deleted through the “Remove attachments” action. If this key is not defined the removed attachments will be permanently deleted, without having the ability to retrieve them. **NOTE:** In case that multiple vaults are used, then this key should be defined for each vault.			
<vaultName>_L_client_export	string		
The Linux path to be used for the synthesis of SPDRM hyperlinks that targets to the exported data structure of the <vaultName> vault. **NOTE:** In case that multiple vaults are used, then this key should be defined for each vault.			
<vaultName>_W_client_export	string		
The Windows path to be used for the synthesis of SPDRM hyperlinks that targets to the exported data structure of the <vaultName> vault. **NOTE:** In case that multiple vaults are used, then this key should be defined for each vault.			
allowEmptySpace_inFilename	boolean	true, false	true
This option controls whether the white space character is allowed, or not to the name of a file or folder that is imported to SPDRM. **NOTE:** If both keys "*regexValidForFilesFoldersImport*" and "*allowEmptySpace_inFilename*" exist in *taxis.conf*, the latter is ignored.			
allUserRoles	boolean	true, false	false
A SPDRM user can be assigned to multiple-roles, with different access rights. By default a user is able to access data that are accessible only by his/her current role. If this parameter is set to “true” , then the user is able to access data that are accessible by any of his/her assigned roles.			

Key	Type	Accepted values	Default value
ansa_auth_type	string	SPDRM, LDAP	SPDRM
This option defines if ANSA/META will login to the SPDRM Server using SPDRM, or LDAP authentication. When this value is set to “LDAP”, ANSA/META sessions are forced to login to the SPDRM Server through the LDAP authentication.			
ansa_multi_output	boolean	true, false	false
This option control whether the SPDRM Server will allow, or not the saving of subsystems –through ANSA- that are defined by multiple includes files. When this option is enabled (true) a subsystem can be defined by its primary/main file, along with several secondary/additional files (attached to main).			
asyncRemoteFileTransfer	boolean		false
When this value is set to true, any file transfer from a remote site to the main site is performed asynchronously. Otherwise, the function/transaction concludes while awaiting for the file transfer to be completed.			
chownBasedOnSPDRMUser	boolean	true, false	false
When the “securityEnabled” key is set equal to “true” all the NodeExec directories that are created by the SPDRM Server, to facilitate the files I/O in the vault, must be accessible only by the specific user id. Thus, the “chown” operation is used by the SPDRM Server in order to change the ownership of these directories and their contents. By default the SPDRM Server performs the “chown” operation based on the OS Client username. When this key is set to “true” the SPDRM Server performs the “chown” operation based on the SPDRM Client logged-in username. Please refer to the Appendix B of the SPDRM Installation Guide for more details about the SPDRM security policy.			
client_acquires_license	boolean	true, false	false
This option controls whether the SPDRM Client application will acquire the client license directly from the BETA License Server (true), instead of acquiring it through the SPDRM Server (false). When this option is enabled (true) it offers the following benefits: • Monitor the users (and the host) that occupy credits of the SPDRM_CLIENT license feature (through the command: beta_lm_stat -a). • Each SPDRM Client will be able to acquire license from the local license server (in case of multi-site deployments).			
componentsContainer	string		Entities
Defines the name of the data container, under which all the model entities (i.e. Subsystems and Parts) will be stored upon their creation.			
default_privileges_group	string		mvx
This key defines the default privileges (e.g. mvdx) that will be granted –upon the creation of an object in SPDRM- to the current group (role) of the user who created the object. The default privileges are granted for newly created objects, if they are not explicitly defined otherwise. The value is defined by any combination of the letters **m** (modify), **v** (view), **d** (delete), and **x** (execute).			
default_privileges_others	string		vx
This key defines the default privileges (e.g. vx) that will be granted –upon the creation of an object in SPDRM- to all the other groups (roles), except for the current group of the user who created the object and its parent groups. The default privileges are granted for newly created objects, if they are not explicitly defined otherwise. The value is defined by any combination of the letters **m** (modify), **v** (view), **d** (delete), and **x** (execute).			

Key	Type	Accepted values	Default value
default_privileges_samesubgroups	string		
This key defines the default privileges (e.g. mvdx) that will be granted –upon the creation of an object in SPDRM- to all groups and recursive subgroups, which share the same parent group with the user who created the object. The default privileges are granted for newly created objects, if they are not explicitly defined otherwise. The value is defined by any combination of the letters **m** (modify), **v** (view), **d** (delete), and **x** (execute).			
default_privileges_subgroups_commongroup	string		
This key defines the default privileges (e.g. vx) that will be granted –upon the creation of an object in SPDRM- to all groups and recursive subgroups, which share the same top-level group with the user who created the object. The default privileges are granted for newly created objects, if they are not explicitly defined otherwise. The value is defined by any combination of the letters **m** (modify), **v** (view), **d** (delete), and **x** (execute).			
delete_time_frame_days	Integer		30
Defines the time period in days that deleted data will remain in Search Deleted list. Clean up mechanism will erase each day all data that have overcome this time period. Value -1 can be used to clean up all data immediately after their deletion.			
DMEntity	string		Subsystem
The name of the subsystem entity as this is defined in the data model structure (*dm_structure_TBM.xml*).			
dmfilters_resultset_limit	integer		4000
This option defines the maximum number of returned results when direct filtering of Data Tree is used.			
dMLIBRARY	string		LIBRARY_ITEMS
Defines the name of the data container, under which all the library items will be stored.			
DMLoadcase	string		Loadcase
The name of the loadcase entity as this is defined in the data model structure (*dm_structure_TBM.xml*).			
DMRun	string		Simulation_Run
The name of the simulation run entity as this is defined in the data model structure (*dm_structure_TBM.xml*).			
DMReport	string		Simulation_Run
The name of the report entity as this is defined in the data model structure (*dm_structure_TBM.xml*).			
DMSession	string		Simulation_Run
The name of the session entity as this is defined in the data model structure (*dm_structure_TBM.xml*).			
DMSimulationModel	string		Simulation_Model
The name of the simulation model entity as this is defined in the data model structure (*dm_structure_TBM.xml*).			

Key	Type	Accepted values	Default value
dmStructureContainer	string		Structure
Define the name of the data container, under which all the simulation data (e.g. Simulation Models, Loadcases, Runs) will be stored in a tree structure way.			
dmTMP	string		TMP
Define the name of the data container, under which all the temporary data that will be created during the workflows execution will be stored.			
enable_permanent_delete_of_files	boolean	true, false	false
This option controls if the files, associated to data objects that are deleted from SPDRM, will be also deleted from the file system (vault), or not.			
enableAddEntities	boolean	true, false	true
This option controls whether the "Add Subsystem" / "Add Part" options will be visible, or not, through the context menu on the "Entities" sub-container (i.e. "Subsystems", "Parts").			
externalFFT	boolean	true, false	false
The default method of the SPDRM server for transferring files between sites (in multi-site deployments) is the scp. However, SPDRM offers the possibility to use 3rd-party high-speed file transfer solutions (e.g. Aspera IBM) instead of scp for transferring files between the main and the remote vaults. Set this value to “true” to enable the use of 3rd-party high-speed file transfer solution. When it is enabled, the keys *externalFFTDownloadFromRemotePattern* and *externalFFTUpload2RemotePattern* should be also properly defined.			
externalFFTDownloadFromRemotePattern	string		
The command of the preferred 3rd-party high-speed file transfer solution for downloading files from a remote file server. Example for the Aspera IBM tool: `ascp -T -l 200m -i [Path_To_SSH_Key] [RemoteSite_SSHhost]@[RemoteSite_SSHuser]:{0} {1}`			
externalFFTUpload2RemotePattern	string		
The command of the preferred 3rd-party high-speed file transfer solution for uploading files to a remote file server. Example for the Aspera IBM tool: `ascp -T -l 200m -i [Path_To_SSH_Key] {0} [RemoteSite_SSHhost]@[RemoteSite_SSHuser]:{1}`			
file_io_microservice_address_<siteName>	string		
The IP address of the machine where the RIOC service is installed for the respective site.			
file_io_microservice_port_<siteName>	integer		
The port that is used by the SPDRM Server for the communication with the RIOC service in the <siteName> site.			
file_transfer_max_parallel_size	intger		
Defines the maximum number of files that can be transferred simultaneously between main and remote site by the RIOC service. The default value is equal to the number of CPU cores of the SPDRM server’s host machine.			

Key	Type	Accepted values	Default value
getVaultPathDelay	Integer		0
This attribute defines time in milliseconds that server will sleep until client responds that a folder or a file has been created. This functionality has been added because of Windows SMB caching.			
http_host	String		
This is the hostname, or IP address of the SPDRM Server machine, as it is accessible by the client applications that performs HTTP requests (e.g. ANSA/META).			
http_port	Integer		8080
This is the port of the SPDRM Server machine that is used for receiving HTTP requests.			
LBREnabled	boolean		false
This option activates lifecycle management of DM objects. A set of rules that govern the evolution of a DM object can be specified for all the available data types, through the “LBR_rules.xlsx” which is located in the SPDRM server configuration folder.			
LBRLoggingEnabled	boolean		
This option activates logging of lifecycle management mechanism. The default folder for the log files is `../server/wildfly/standalone/log/spdrm/LBR`			
LDAP_account	string		
The user DN of a technical/administrator LDAP/Active Directory account that is able to search users in LDAP/Active Directory server. This is required in cases that the LDAP User DN pattern is not standardized / known for all users (e.g. in case that users belong to various organization units). e.g. `uid=admin,ou=People,dc=localdomain`			
LDAP_account_pwd	string		
The password of the above technical/administrator LDAP/Active Directory account.			
LDAP_dn_attribute	string		
Defines the key which LDAP server recognizes as username field. This value is usually *cn* or *uid*.			
LDAP_filter	string		
The filter that will be used for the search of the User DN. e.g. `(&(uid={0})(objectClass=organizationalPerson))`			
LDAP_URL	string		
The URL of the LDAP/Active Directory server. This key is required, along with the *LDAP_username*, in order to integrate LDAP/Active Directory server to SPDRM (for user authentication, and fetching users through LDAP/Active Directory). E.g. ldap://ldap.localdomain:389 Multiple LDAP URLs can be defined that will be used in a Round Robin format by the SPDRM Server in order to find the first available. URLs must be semicolon separated. e.g. `ldap://ldap.localdomain:389; ldap://ldap2.localdomain:390`			

Key	Type	Accepted values	Default value
LDAP_username	string		
The User DN pattern of the LDAP/Active Directory server. This key is required, along with the *LDAP_URL*, in order to integrate LDAP/Active Directory server to SPDRM (for user authentication, and fetching users through LDAP/Active Directory). e.g. `uid={0},ou=sup,ou=People,dc=localdomain`			
LDAP_username_RR			
The User DN pattern of the LDAP/Active Directory server in Round Robin format. This key is required, along with the *LDAP_URL*, in order to integrate LDAP/Active Directory server to SPDRM (for user authentication, and fetching users through LDAP/Active Directory) when multiple LDAP usernames must be defined. The SPDRM Server will use them in a Round Robin format. If both LDAP_username and LDAP_username_RR are defined, the second one will prevail. Usernames must be semicolon separated. e.g. `uid={0},ou=sup,ou=People,dc=localdomain; uid={0},ou=dev,ou=People,dc=localdomain`			
LibraryItems_targetVault	string		
This option enables the system administrator to globally define the target vault during importing library items. This option is applicable in SPDRM installations with multiple vaults, when there is a need to store all library items under a specific vault (e.g. vault1).			
loadcaseHeaderRLI	string		-
The name of the loadcase header RLI as is defined in the data model (*dm_structure_TBM.xml*)			
loadcaseTemplateRLI	string		Loadcase_Template
The name of the loadcase template RLI as is defined in the data model (*dm_structure_TBM.xml*)			
mxnInstanceLimit	integer		20
Defines the limit that control the maximum number of MxN instances that run simultaneously.			
noGuiClientPath	string		
The full path to the SPDRM No-GUI Client start script.			
noGuiClientSshHost	string		
The hostname, or IP address of the machine that hosts the SPDRM No-GUI Client sessions. If set, the SPDRM Server connects through SSH to that machine, otherwise it start the SPDRM No-GUI Client application as a local process.			
noGuiRoot	boolean	true,false	true
This option controls if the no-gui client will be executed by the logged-in user as root by using 'su' or not. The key is valid if only the value of the **no_gui_mode** key is set to **local**.			
no_gui_mode	string	local,ssh,remote	local
The mode on which SPDRM operates in order to run the no-gui client: • **local**: SPDRM Server tries to start the NoGUI session on local machine. • **ssh**: SPDRM Server tries to start the NoGUI session on a machine defined in the *noGuiClientSshHost* key, through SSH. • **remote**: SPDRM Server tries to start the NoGUI session through BETA Apps Launcher.			

Key	Type	Accepted values	Default value
noGuiPollTimeoutSeconds	integer		120
The maximum duration (in sec) that the SPDRM Server waits for the SPDRM No-GUI Client to login.			
regexValid	string		
Any valid regex expression can be set to this key to control the allowed characters on the meta-data text fields of data objects that are going to be imported to SPDRM. e.g. `^[A-Za-z0-9*${}._:-]+$`			
regexValidForFilesFoldersImport	string		
Any valid regex expression can be set to this key to control the allowed characters on the name of files and folders that are going to be imported to SPDRM. e.g. `^[A-Za-z0-9*${}._:-]+$` **NOTE:** If both keys "*regexValidForFilesFoldersImport*" and "*allowEmptySpace_inFilename*" exist in *taxis.conf*, the latter is ignored.			
set_target_vault_per_role	boolean	true, false	false
It controls the setup of a default target vault per role. This option applies to SPDRM installations with multiple vaults. After setting a target vault to a role, all data that are saved/imported by users with this particular role are automatically stored in the selected vault. By default this option is disabled. Set this value to “true” to enable it. After enabling it, the option “Set Target Save Vault” is accessible through the context menu of roles, in the “Users Management” tool.			
simulationConfigurationTableRLI	string		Configuration_Table
The name of the configuration table RLI as is defined in the data model (*dm_structure_TBM.xml*)			
sites	string		
This key is used to define the name of the sites (comma separated value). It is applicable only for SPDRM multi-site installations.			
skip_mxn_instances	boolean	true, false	false
This option controls whether to skip, or not, failed MxN instances and continue to the execution of the next node of the workflow. It applies to MxN nodes with “Execution Mode” set as “Sequential”. By default the process stops its execution in case at least one MxN instance failed at runtime.			
spdrm_auth_type	string	SPDRM, LDAP	SPDRM
This option defines if SPDRM Client will login to the SPDRM Server using SPDRM, or LDAP authentication. When this value is set to LDAP, then the SPDRM Client is forced to login to the SPDRM Server through the LDAP authentication.			
sshLog	boolean	True,false	false
This option controls the creation of a file that logs all SSH requests from a remote site to main site and vice versa. When this option is enabled (true) a file (ssh.log) is generated in the SPDRM log directory (alongside spdrm.log)			
workingDirSLP	integer		0
This parameter is used by the mechanism on the SPDRM Server that bypasses the SMB client side caching when accessing files in the SPDRM vault. It defines the duration (in msec) that the SPDRM Server will sleep/wait until informing the SPDRM Client for the creation of the folder/file on the vault.			

Key	Type	Accepted values	Default value
workingDirSLP_WS	integer		0
This parameter is used by the mechanism on the SPDRM Server that bypasses the SMB client side caching when accessing files in the SPDRM vault. It defines the duration (in msec) that the SPDRM Server will sleep/wait until informing the ANSA/META for the creation of the folder/file on the vault.			
wsTimesLog	boolean	true, false	false
This option controls the creation of a file that logs the duration (response time) of web service calls (i.e. ANSA/META requests through HTTP). When this option is enabled (true) a file (*wsTimes.log*) is generated in the in the root level of the default vault.			

3.7. SPDRM Client Configuration

3.7.1. Required settings on taxisprops.xml

The configuration settings of the SPDRM Client are included in the ***taxisprops.xml.linux*** and ***taxisprops.xml.windows*** files that are located in: [SPDRM_CLIENT_DIR]/.

IMPORTANT: The directory paths that are used in these files should always end with a trailing slash.

Key	Type	Accepted values	Default value
<vaultName>	string		vault1
The full path to the <vaultName> vault, as it is accessible by the client machine. **NOTE:** In case that multiple vaults are used, then this line should be defined for each vault.			
export_<vaultName>	string		export_vault1
The full path to the exported data structure of the <vaultName> vault, as it is accessible by the client machine. **NOTE:** In case that multiple vaults are used, then this line should be defined for each vault.			
dmitem_icons_path	string		[SPDRM_CLIENT_DIR]/dmicons/
The full path to the parent folder of the icons used by the DM items, as it is accessible by the client machine. **NOTE:** The DM icons are defined in the data_views.xml using relative path to this directory.			
serverAddress	string		localhost
The hostname, or IP address of the SPDRM server machine.			
serverPort	integer		8080
The port for the Client-Server communication.			
serverVendor	string	jboss, wildfly, jboss_cluster_standalone, jboss_cluster, weblogic, weblogic_ssl, glassfish	wildfly
Defines explicitly which client configuration of the specified application server will be used.			

3.7.2. Optional settings on taxisprops.xml

Key	Type	Accepted values	Default value
auto_login	boolean	true, false	false
Defines explicitly whether the user is able to automatically login to the SPDRM client without prompting for credentials, or not. If this key is set to true, the SPDRM Client uses the OS username in order to authenticate to the SPDRM Server. **NOTE:** The auto login requires that a user with the same name as the OS user has been already created in SPDRM.			
default_auth_type	string	SPDRM, LDAP	SPDRM
Defines explicitly the authentication type for the SPDRM Client. • The **SPDRM** authentication requires the setup of a user password in SPDRM, which is stored encrypted in the SPDRM database. • The **LDAP** authentication does not require the setup of a user password in SPDRM. However, it requires the creation of the user in SPDRM, and the proper LDAP configurations.			
display_name	string		[User Name]
Defines the display name of the SPDRM user within the different user-related fields / lists. The value of this key should be an expression that makes use of any (or a combination) of the following SPDRM fields/parameters: [First Name], [Last Name], and [User Name].			
enable_weblogic_auth	boolean	true, false	false
Set this option to “true” if the WebLogic authentication is required. **NOTE:** If this option is set to “true”, then the “weblogic_username” and “weblogic_password” keys should be also properly defined.			
externalPathOsMapping	dictionary		
This parameter is used by the External File System functionality in case there is different OS between the SPDRM Server and the SPDRM Clients. It is declared as a dictionary where the key is the SPDRM Server's path while the values is the SPDRM Client's path.			
file_seeker_interval	integer		100
This parameter is used by the mechanism on the SPDRM Client that bypasses the SMB client side caching when accessing files in the SPDRM vault. It defines how often the Windows SPDRM Client will search for a file on a SAMBA share network path (in msec).			
file_seeker_timeout	integer		3000
This parameter is used by the mechanism on the SPDRM Client that bypasses the SMB client side caching when accessing files in the SPDRM vault. It defines how long the Windows SPDRM Client will wait for the existence of the file on a SAMBA share network path (in msec).			
file_transfer_timeout	integer		300
This parameter is used by the SPDRM remote client to define the waiting time for a file transfer from the main site to remote site. The timeout is defined in seconds.			
HistoryNodedistance	integer		50
Defines the distance of two connected objects (length of connection line) in the Lifecycle Graph.			

Key	Type	Accepted values	Default value
idle_user_timeout	integer		0
The SPDRM Client could automatically log out and release the occupied license when a user is inactive for more time than a time-out limit. By default this mechanism is disabled. To enable it an integer value greater than 0 should be specified in minutes to this key. **NOTE**: User inactivity is defined according to the following rules: • No mouse clicks of the user on the SPDRM Client. • No running nodes, executed by the user. • No transactions running on the server, triggered by the user. • No scheduled nodes, assigned to the user. • No locked* file by the user. (*) A file is marked as "locked" when a user selects to Edit the file. The "lock" status of a file is reset when the user exits the editing application.			
ldapMapping	json string		{FirstName:givenName,LastName:sn,UserName:uid,Email:mail}
This parameter is used by the mechanism that fetches LDAP users in SPDRM. It defines a map (in json format) between the LDAP user attributes and the SPDRM user profile fields e.g. {FirstName:givenName,LastName:sn,UserName:uid,Email:mail}			
log_script_functions	boolean	true,false	false
When value is set as true, all the script function calls print a log entry with the name of the function and the execution time.			
script_nodes_limit	integer		20
Defines how many script nodes can be executed concurrently on the SPDRM Client.			
script_process_port	Integer		5000
By this parameter, the user can explicitly define the communication port of SPDRM Client with script launcher.			
site	string		
This parameter is applicable only on the configuration of a remote SPDRM Client, in a SPDRM multi-site installation. It defines the site of the remote SPDRM Client, in order for the SPDRM server to be aware of the origin of the client request. **NOTE**: The value of the site should be one of the available sites, which are defined in taxis.conf.			
user_script_cache_size	float		-1
Defines the cache size (in GBytes) that can be used in order to cache script files that are downloaded from DM, during the execution of script actions. By default this option is disabled (-1). To enable it a float value greater than 0 should be specified in GBytes to this key.			
user_script_cache_path	string		<user_tmp_path>
Defines the directory that can be used in order to cache script files that are downloaded from DM, during the execution of script actions. By default this directory is the temporary directory of the user depending on the SPDRM Client's OS.			
weblogic_password	string		weblogic1
The password of the account that will be used for the authentication to the WebLogic server (if WebLogic authentication is required).			

Key	Type	Accepted values	Default value
weblogic_protocol	string	http, t3	t3
The protocol used for the communication between the SPDRM Client and the WebLogic Server.			
weblogic_ssl_protocol	string	https, t3s	t3s
The SSL protocol used for the communication between the SPDRM Client and the WebLogic Server.			
weblogic_username	string		weblogic
The username of the account that will be used for the authentication to the WebLogic server (if WebLogic authentication is required).			
wildfly_protocol	string	http-remoting, https-remoting	http-remoting
The SSL protocol supported by the Wildfly (orJBoss) application server.			

3.7.3. Enable Debug Log Messages

There are different ways to log messages on the SPDRM Client, in order of severity:

- FATAL
- ERROR
- WARN
- INFO
- DEBUG
- TRACE

By default the log level of the log file of the SPDRM Client (i.e. *taxis.log*) is set to INFO.

This means that the log file contains messages of log levels: FATAL, ERROR, WARN, and INFO.

However, in case of debugging it is useful to get also the messages of log level: DEBUG.

To enable debug log messages on the log file of the SPDRM Client the following modification is required in the line 25 of the *log4j.properties* (located in [SPDRM_CLIENT_DIR]):

```
25  log4j.rootLogger=DEBUG, A1, A2
```

3.7.4. User settings location/Custom label

3.7.4.1. Configuration of the user settings directory

The ***SPDRM_USER_DIR*** variable in the SPDRM Client's start scripts (**startClient.sh** and **startClient.bat** for Linux and Windows OS respectively, as well as the respective NoGUI start scripts - **startClient-nogui.sh** and **startClient-nogui.bat**), defines the location of the user settings.

The default directory for the user settings in the default start script is:

```
%HOME%\.BETA\SPDRM\%ComputerName%\version_%SPDRM_VERSION%\%username%_%ComputerName%
```

The default user settings directory is constructed by combining SPDRM and system information. By changing that variable in both scripts, a custom user settings directory can be defined according to administrator's needs.

3.7.4.2. Configuration of custom label

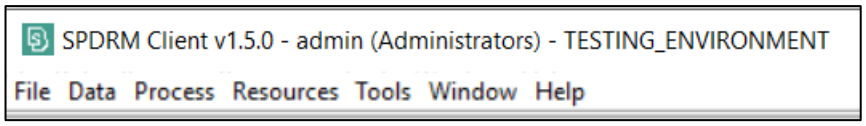
Each SPDRM Environment can be labeled in order to be distinguished by the SPDRM Client used. The label can be defined in all SPDRM Client's start scripts (**startClient.sh** and **startClient.bat** for Linux and Windows OS respectively, as well as the respective NoGUI start scripts - **startClient-nogui.sh** and **startClient-nogui.bat**) by adding the following argument in the client execution line:

```
-J-Dspdrm-label=CUSTOM_LABEL
```

The execution line is the next one of the "*echo.SPDRM client starting...*" , and by adding a custom label will have the following format:

```
"%SPDRM_PATH_TO_EXE%\taxisnetbeans64.exe" --jdkhome %JAVA_HOME% --userdir "%SPDRM_USER_DIR%" -J-Duser.dir="%SPDRM_USER_DIR%"
-J-Dlog4j.configuration=file:"%SPDRM_CLIENT_DIR%\log4j.properties" -J-Dtaxis.props.location.windows="%SPDRM_CLIENT_DIR%\taxisprops.xml.windows"
-J-Dspdrm-label=TESTING_ENVIRONMENT -J-Dlink.command=%1
```

The label (TESTING_ENVIRONMENT) defined in the example script will appear in the title bar of the SPDRM Client.

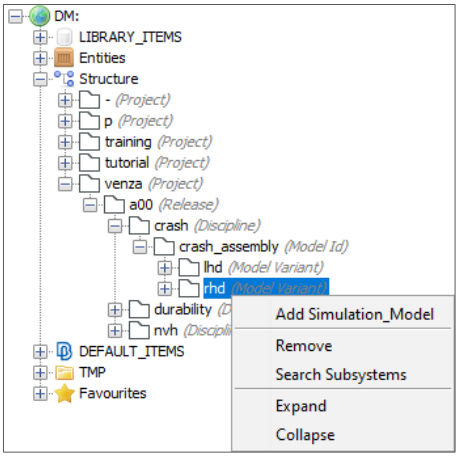


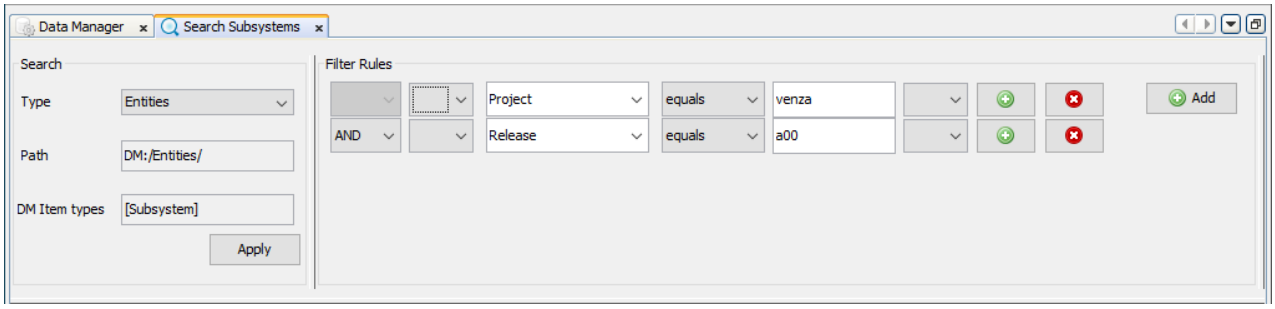
3.7.5. Filter Rules Mapping Configuration

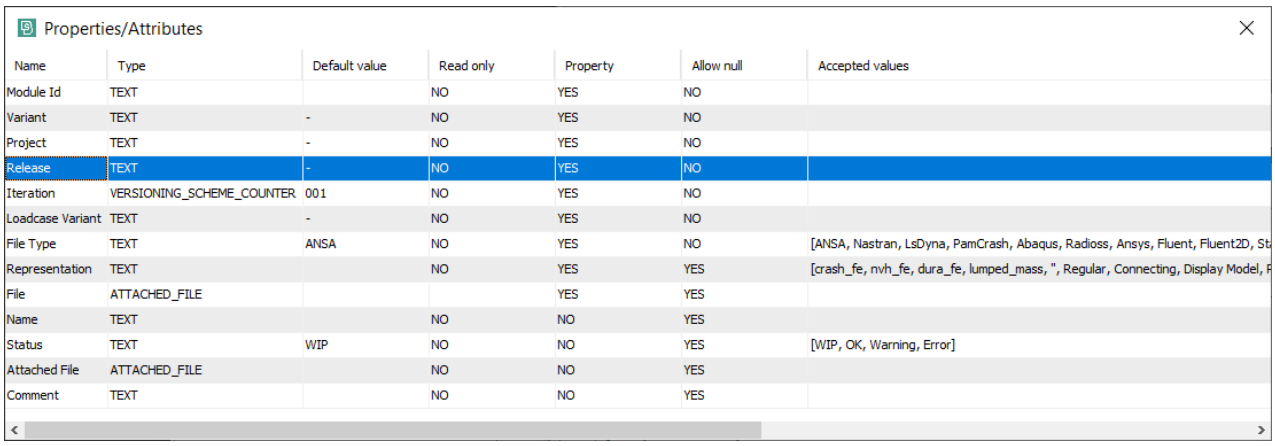
A Search Subsystem option can be found in any virtual folder in the Data Tree. This particular option enables the searching of Subsystems in the database based on a specific level on the Structure.

When the option is used the Search DM window with pre-filled rules based on the level in the Data Tree opens. If no filter rule mapping exists, those rules will follow the 1-to-1 correspondence between the Simulation Model and the Subsystem properties.

The images below show that only the Simulation Model's properties **Project** and **Release** directly match properties of the Subsystem object.



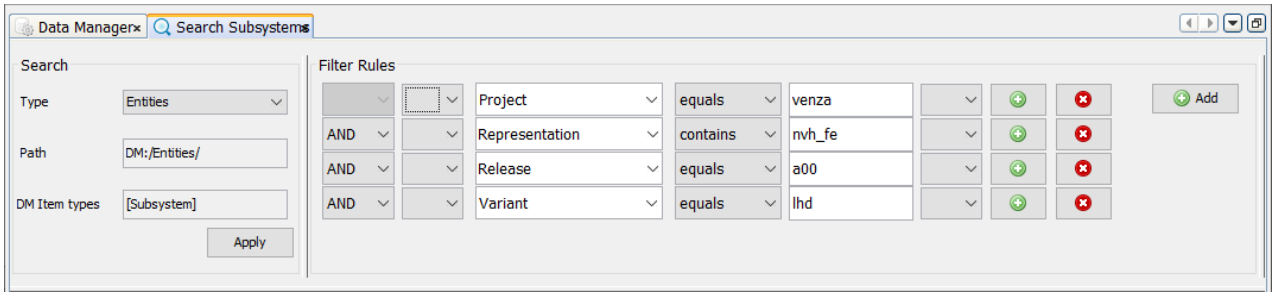




A special file named *mapDMItemProps.json* must be created in the SPDRM server configuration file that will provide mapping between DM objects by using all available comparison operators (equals, contains, etc) for filtering purposes.

The image on the right shows a sample of a ***mapDMItemProps.json*** file, while the image below shows the filter rules that will be applied to the initial scenario after the appliance of filtering mapping. **Representation** and **Variant** are now added.

```
{
  "Simulation_Model": {
    "Model Variant": {
      "Subsystem.Variant": "equals"
    },
    "Discipline": {
      "Part.Module Id": "contains",
      "Subsysten.Representation": "contains"
    }
  }
}
```



4. Data Management

4.1. Data deletion

4.1.1. Permanently deleted objects

Objects without any references can be deleted permanently in SPDRM. To prevent accidental loss of data, it is now possible to add a configurable time frame to delay the deletion of the data from the system.

This is controlled through the *taxis.conf* using the *delete_time_frame_days* key:

```
<entry key="delete_time_frame_days">number_of_days</entry>
```

This way, deleted data will always end up in the *Search Deleted* list, from where the system will automatically remove them after a specified number of days (a "cleaning" mechanism is executed nightly, removing objects with a deletion due date tag equal to the mechanism execution date).

(e.g. for data to be permanently removed from the system 2 days after their deletion by a user, the following entry should be added to the taxis.conf*:* `<entry key="delete_time_frame_days">2</entry>`)

NOTE: If the delete_time_frame_days key is missing from the *taxis.conf,* by default SPDRM will automatically delete the data after 30 days.

NOTE: If a negative value is given for the **delete_time_frame_days** key (i.e. -1), objects that can be permanently deleted will be directly removed from the system, without ending up in the *Search Deleted* list.

For objects that are permanently deleted, SPDRM by default removes their database entries, but not the actual files from the file system (firstDir). This behavior can also be controlled through the *taxis.conf* using the *enable_permanent_delete_of_files* key:

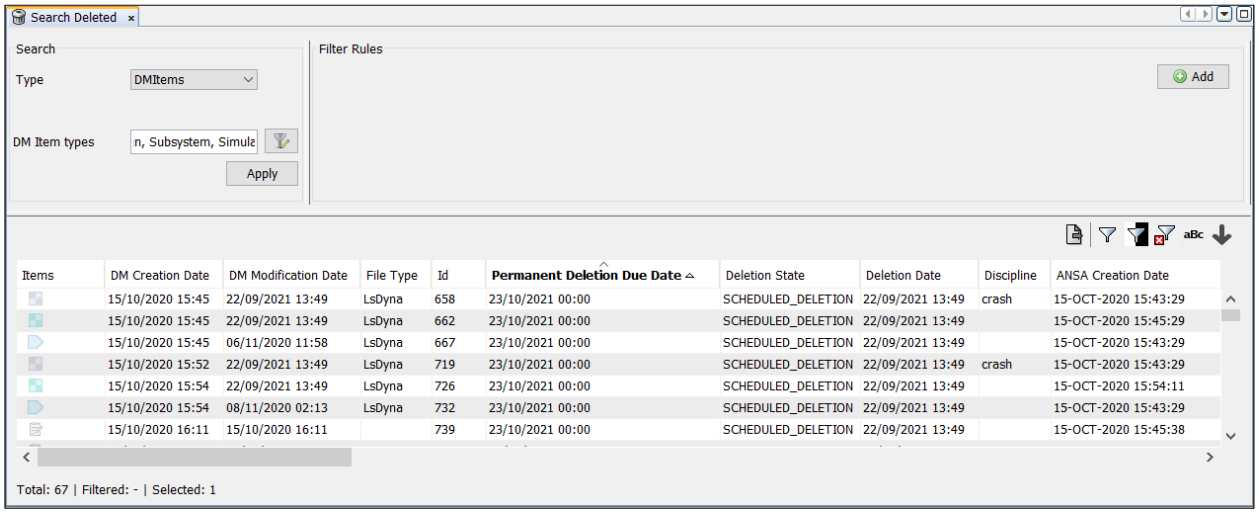
```
<entry key="enable_permanent_delete_of_files">boolean_value</entry>
```

This way, depending on the value (true/false) of the *enable_permanent_delete_of_files* key, deleted data can also be deleted from the firstDir.

NOTE: If the **enable_permanent_delete_of_files** key is missing from the *taxis.conf,* by default SPDRM will **not** delete the data from the firstDir.

4.1.2. The Search Deleted list

The **Search Deleted** list is a tool that was developed to offer functionality related to the management of deleted objects and the data storage. It is a dedicated tool, accessible only by system administrators. It is accessed use the option **Data > Search Deleted** from the SPDRM menu bar.



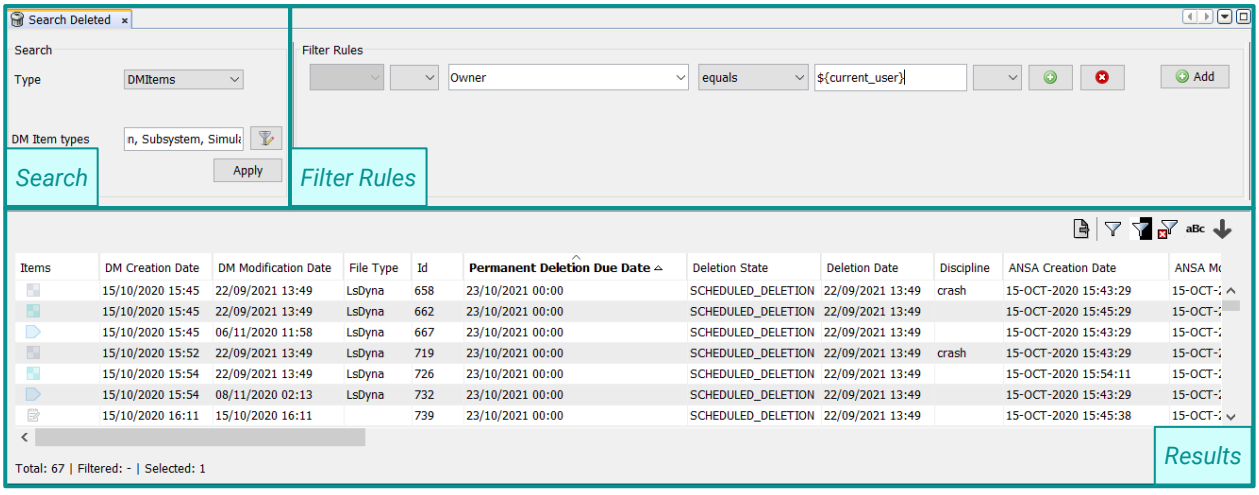
Key features of the Search Deleted tool are:

- It enables the restoration of deleted objects [1]
- It enables the manual purging of deleted objects [2]
- It offers dedicated views for the References, Contents and Lifecycle Graph of deleted objects
- It offers information regarding the deletion of each object (i.e. Deletion User, Deletion Date, etc.)

[1] *Conditions apply (i.e. it would not be possible to restore a Simulation Model if any of its contents have been deleted)*

[2] *Conditions apply (i.e. it would not be possible to manually purge a partially deleted object, if any of its descendants are still "live")*

The *Search Deleted* list has similar search functionality as the *Search DM* window, described in the Data Search section.



In the *Search* section, the user can define the search settings for the execution of filter queries and **Apply** the filters

In the *Filter Rules* section, the user can define the filters for the query execution and **Add** new filters

In the *Results* list, the user can view the results of the applied filter and search within the displayed results

The ***Deletion State*** and the ***Permanent Deletion Due Date*** columns offer information regarding the deleted data and their scheduled removal from the system.

Items	Owner	Size	Creation Date	Version	Id	Deletion State	Permanent Deletion Due Date
	demo	3.3 kB	04/07/2018 16:11	1	51281	SCHEDULED_DELETION	02/09/2018 11:12
	demo	0 B	04/07/2018 16:12	1	51283	OVERWRITTEN	
	s.tzamtzis	1.9 MB	09/03/2018 13:24	1	46713	SCHEDULED_DELETION	02/09/2018 11:12
	s.tzamtzis	26.2 kB	09/03/2018 14:34	1	46761	SCHEDULED_DELETION	02/09/2018 11:12
	s.tzamtzis	77.3 kB	19/09/2017 13:33	1	4369	SCHEDULED_DELETION	02/09/2018 11:12
	demo	44.1 kB	14/06/2018 15:17	1	50763	OVERWRITTEN	
	demo	206.0 kB	14/06/2018 15:17	1	50762	OVERWRITTEN	

The Deletion State column has three possible values:

OVERWRITTEN, for objects that exist as overwrites of existing objects. These objects can be purged manually by an administrator at any time.

HISTORY_LINK, for objects that have history links with other objects. These objects can be purged manually by an administrator only if all their descendants are deleted.

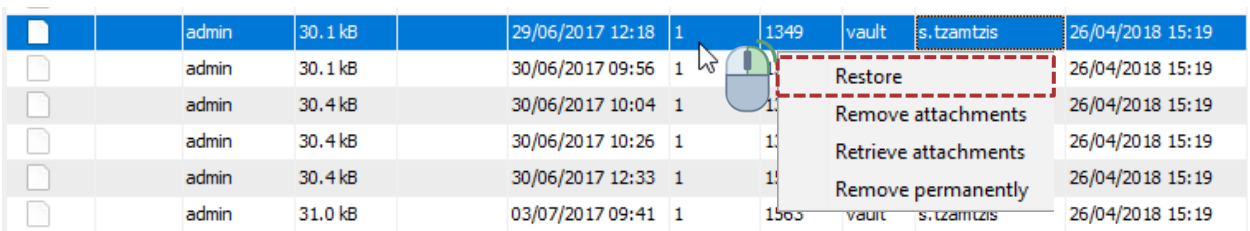
SCHEDULED_DELETION, for objects that will be automatically purged by the system on a scheduled date. These objects can be purged manually by an administrator at any time.

The Permanent Deletion Due Date column displays the exact date when the scheduled purging of an object will automatically be performed by SPDRM.

The ***Deletion Date*** and the ***Deletion User*** columns can assist the administrator in identifying deleted data.

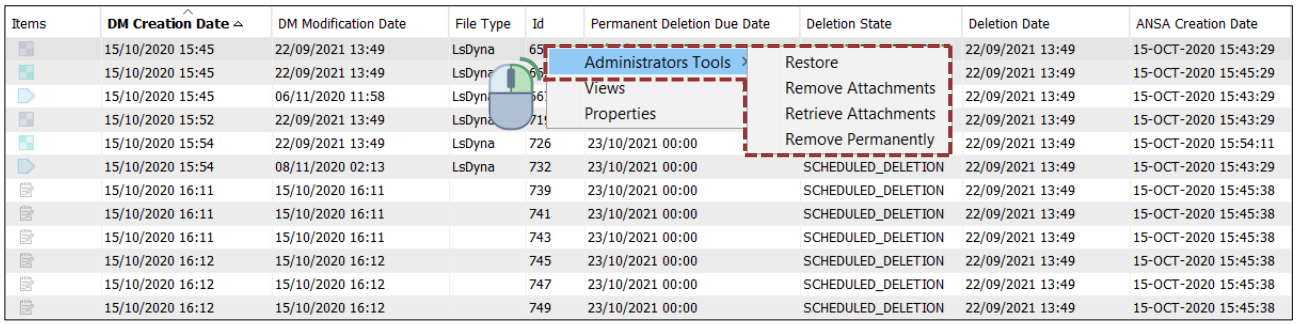
Items	Status	Owner	Size	DM Path	Creation Date	Version	Id	Vault	Deletion User	Deletion Date
		demo	123.6 kB		20/07/2017 18:27	1	3715	vault	demo	14/06/2018 15:17
		demo	124.2 kB		20/07/2017 23:15	1	3838	vault	demo	14/06/2018 15:17
		demo	124.2 kB		21/07/2017 16:56	1	3963	vault	demo	14/06/2018 15:17
		demo	133.4 kB		28/08/2017 12:07	1	4116	vault	demo	14/06/2018 15:17
		demo	134.6 kB		26/09/2017 10:43	1	7942	vault	demo	14/06/2018 15:17
		demo	30.1 kB		29/06/2017 12:18	1	1349	vault	demo	26/04/2018 15:19
		demo	30.1 kB		30/06/2017 09:56	1	1361	vault	demo	26/04/2018 15:19
		demo	30.4 kB		30/06/2017 10:04	1	1362	vault	demo	26/04/2018 15:19
		demo	30.4 kB		30/06/2017 10:26	1	1372	vault	demo	26/04/2018 15:19
		demo	30.4 kB		30/06/2017 12:33	1	1521	vault	demo	26/04/2018 15:19

If the data was accidentally deleted, the administrator can proceed to the restoration knowing who the deletion user was and when the data was deleted.



4.1.3. Removal of attached files/folders

The SPDRM deletion mechanism enables the removal, archival and retrieval of attached files /folders for all DM objects. The defintions of the DM objects and their metadata will remain in the system, while their attachments will be removed to facilitate storage space management. This functionality is available through the respective context menu option, accessible only by system administrators.



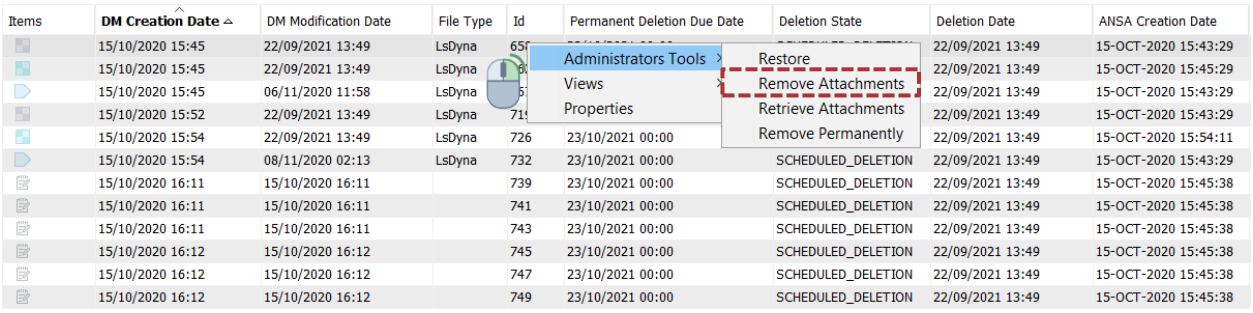
When attachments are being removed, it is possible for SPDRM to archive them in a pre-defined location on the file system, in order to enable their retrieval. This is controlled through the *taxis.conf* using a dedicated key:

```
<entry key="<vaultName>ArchiveDeleted">/....filepath/</entry>
```

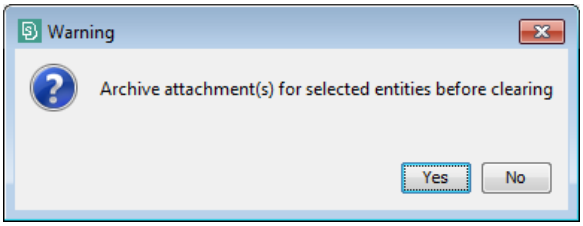
Once an archival path is defined, SPDRM will support the archival of attachments being removed, if requested by the administrator. A *.zip file named after the handle Id of the object whose attachments are being removed, containing all its attachments, will be saved in the archival directory. This can be retrieved on demand, provided that the *.zip file has not been modified after its creation.

NOTE: SPDRM requires the definition of a separate archival path per vault. Therefore, in order to enable the archival functionality, the entry key should be written per vault in the *taxis.conf*.
e.g. to enable the archival functionality for objects stored in a vault named "*vault*" to an archive directory in the filepath such as "*/mnt/archives/SPDRM/vaultArchiveDeleted/*", the following entry should be added:

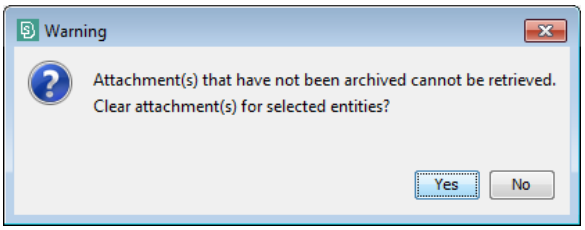
```
<entry key="vaultArchiveDeleted">/mnt/archives/SPDRM/vaultArchiveDeleted/</entry>
```



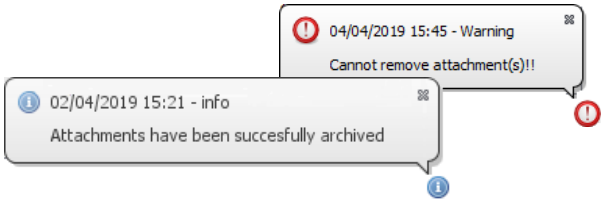
Selecting the option **Remove attachments** for a DM object, an SPDRM warning window will appear asking the user if the attachments should be archived before the removal. **NOTE** that if the archival functionality is not enabled through the *taxis.conf*, this warning window will not appear.



Before removing the attachments of a DM object, an SPDRM window will appear to inform the user that attachments which have not been archived cannot be retrieved if required; and ask for confirmation that the attachments of the selected object(s) should be removed.

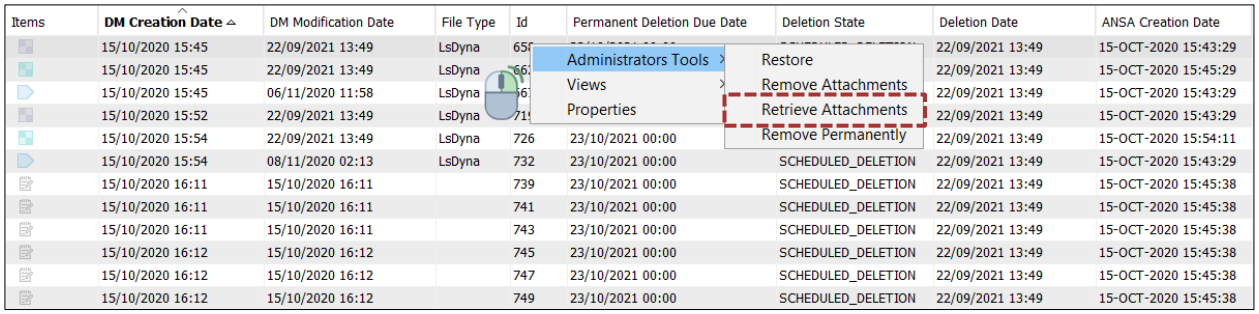


A notification balloon will appear on the notification displayer section of the SPDRM client, informing the user for the success / failure of the removal of the attachments.

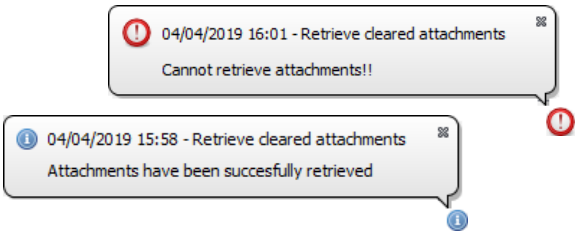


NOTE: It is possible to define custom icons for objects whose attachments have been cleared, in order to facilitate their visual identification in the SPDRM Client.

Selecting the option **Retrieve attachments** for a DM object whose attachments have been removed, SPDRM will look in the archive directory for a .zip file named after the handle Id of the selected object and will try to restore the attachments.



A notification balloon will appear on the client, informing the user for the success / failure of the retrieval of the attachments.



Note that, SPDRM will fail to retrieve the attachments of a DM object if the .zip file containing them does not exist,

or if it has been modified since its creation.

The functionality for the removal/retrieval of attached files/folders is also available via script, using the following SPDRM script functions:

- `dm.clearAttachments(itemid, archive)`
- `dm.retrieveAttachments(handle_ids)`
- `dm.deleteArchivedAttachments(handle_ids)`

Please refer to the SPDRM scripting guide for a detailed description of the script functions.

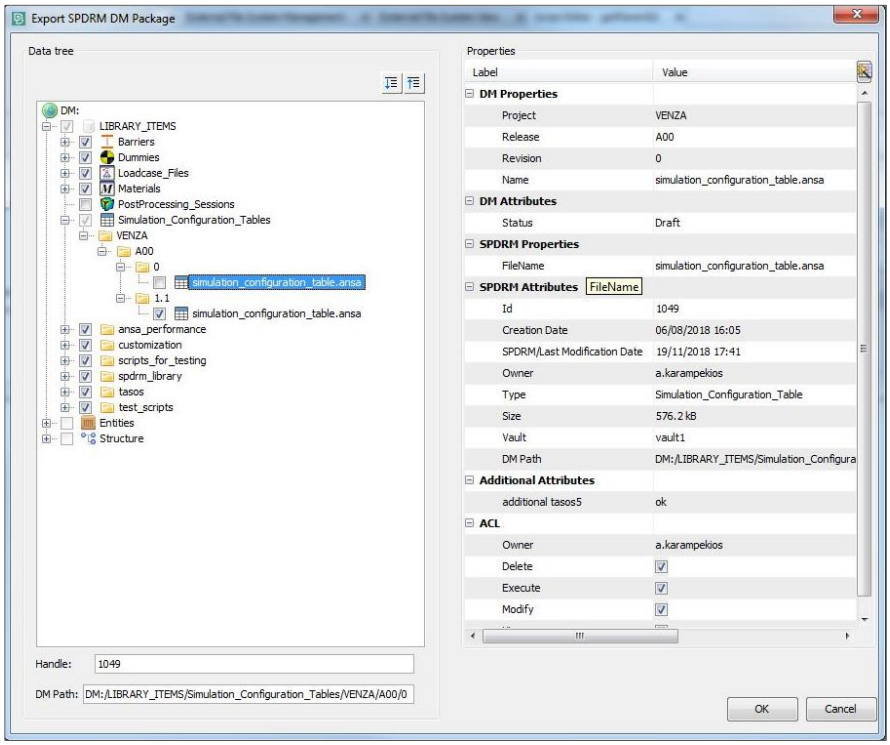
4.2. Data migration

4.2.1. DM Export

DM Export functionality allows an SDPRM Administrator to export massively data from an SPDRM Database in compressed format. The exported .zip file can then be used to import those data in another SPDRM Environment.

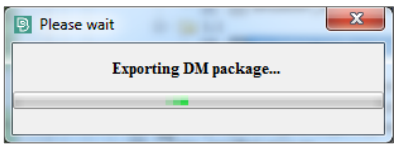
Go to Data > DM Export Import > DM Export to access DM Export functionality. The tool offers flexibility to the selection of the data to be exported by using the Data Tree as shown in the image.

By opening the **Export SPDRM DM Package** window the user has access to all data through data tree and can select the desirable data for export with the respective checkboxes.



After pressing the **OK** button, the user is asked to select the export directory.

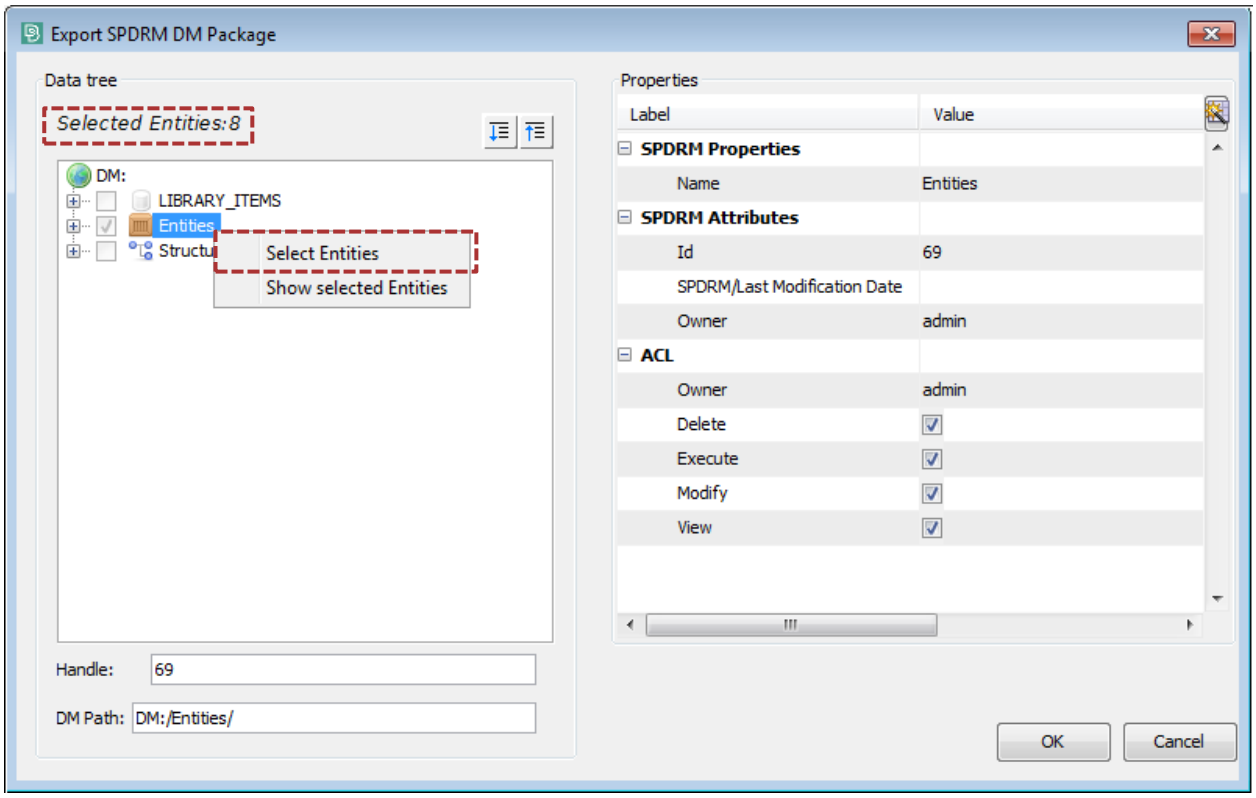
The exporting window appears and the client remains idle as long as the export procedure takes place.



The user must have in mind that the exporting procedure may take long time, depending on the size of the exported data.

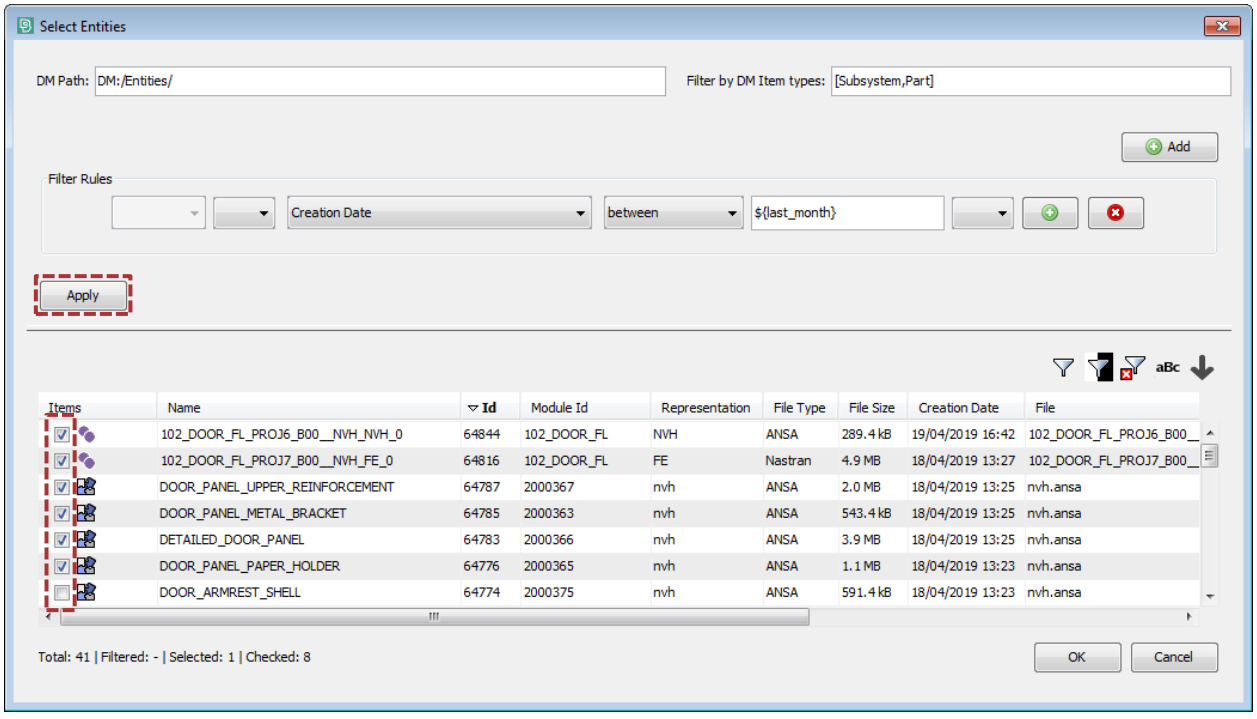
The compressed file is exported in the selected directory, with the exporting date written in the file name.

The **DM Export** tool supports the option to search and select particular entities (Subsystems, Parts) to be exported. The **Select Entities** window is invoked through the context menu of the Entities container.



In the **Select Entities** window a query can be applied to search for the entities that need to be exported.

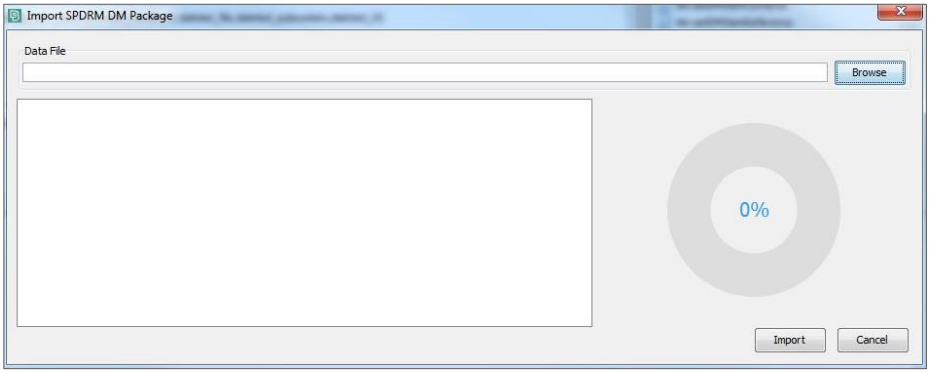
The entities that need to be exported should be marked as such, by enabling the check-box in the first column. By pressing the **OK** button the number of selected entities is displayed in the Export **SPDRM DM Package** window.



4.2.2. DM Import

Reverse process from DM Export is the **DM Import** procedure. **DM Import** can be used for importing data from a compressed file to the SPDRM Database massively.

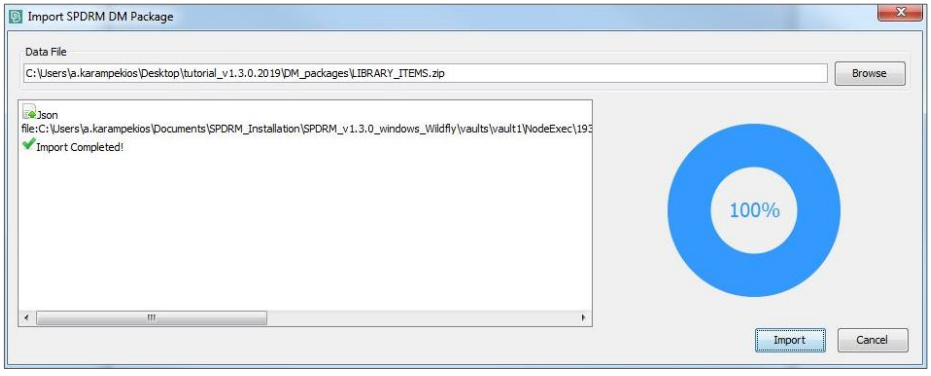
Go to Data > DM Export Import > DM Import to access DM Import functionality. At the start window of the **Import SPDRM DM Package**, the user must select from the file-system the .zip file that will be imported.



After file selection, SPDRM will extract the selected compressed file and will inform the user when extraction has been completed. By pressing Import the user can start the import procedure.

Before starting importing data, SPDRM will search for conflict between existing and incoming data, asking for a user decision.

After the conflict resolution decision, the user can monitor the import procedure from the progress bar. An 'Import Completed!' message will inform the user about the successful termination of the procedure.

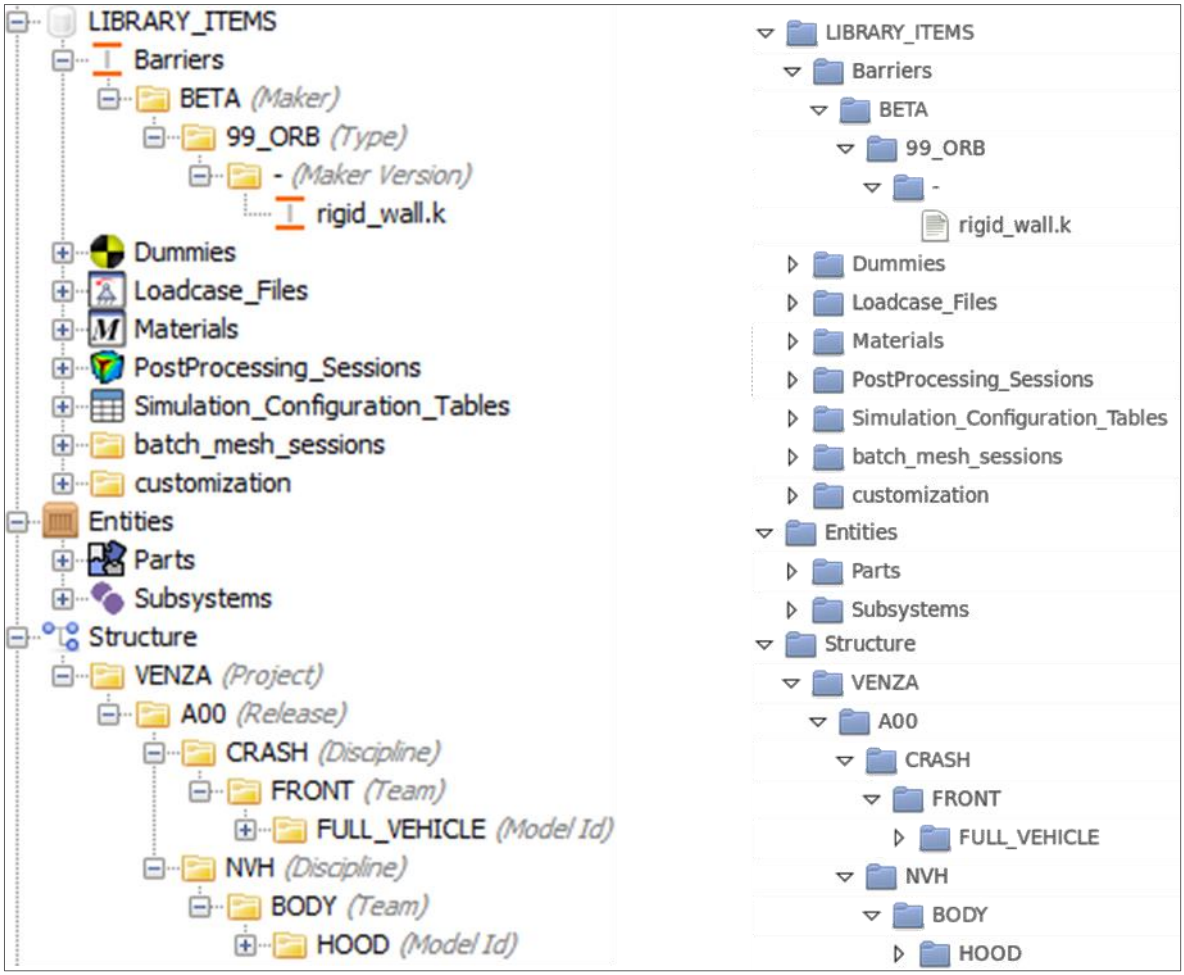


4.3. External data structure

SPDRM support the option to create a read-only file structure on a designated location on the file-system, which is a precise representation of the SPDRM data structure. This file structure is automatically maintained by the SPDRM server in real-time.

The exported file structure provides direct access to files of the SPDRM vaults to external users and applications.

Example:

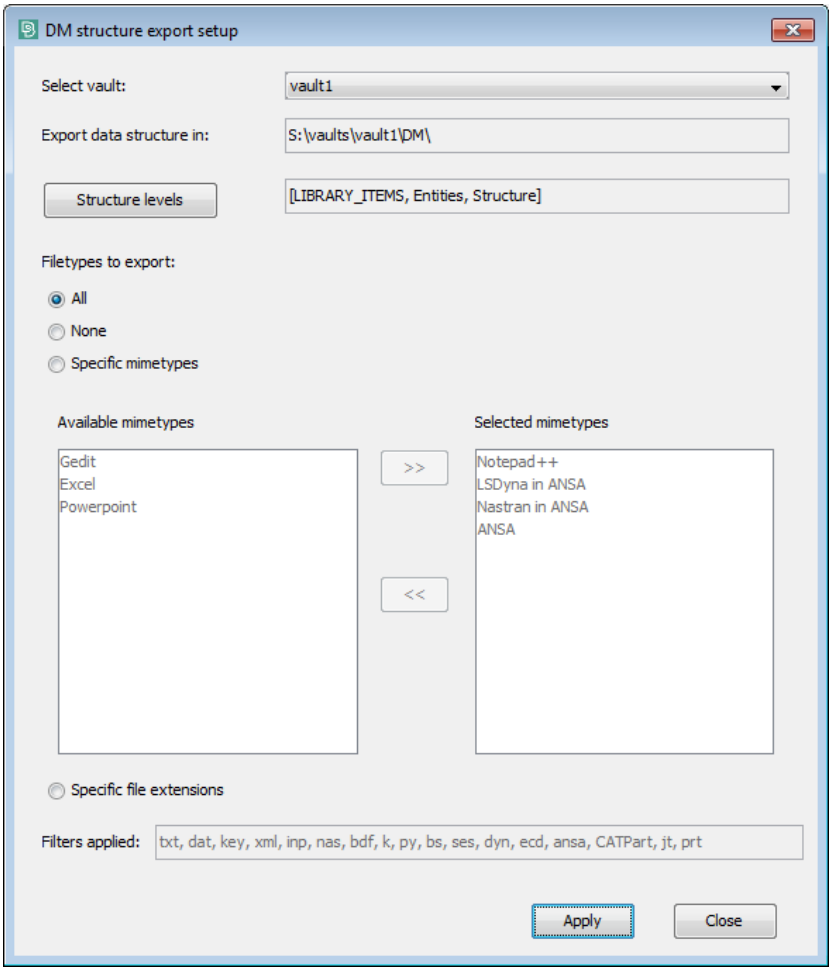


SPDRM DM Tree *Exported file structure*

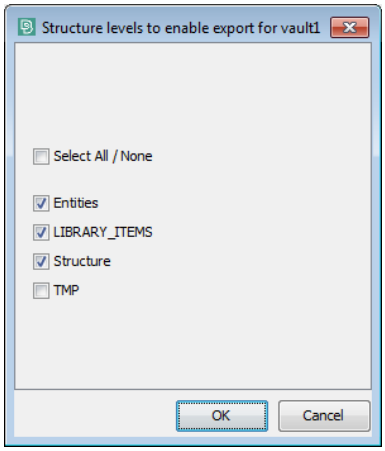
4.3.1. Configuration

By default the mechanism that creates and maintains the exported file structure is disabled. It is activated and configured through the *DM structure export setup* window invoked from **Data > DM structure export setup**.

In case that more than one vaults are defined, the DM structure export setup should be configured for each vault separately, using the **Select vault** drop down menu.



The data structure levels to be exported are defined through the **Structure levels** button.



The exported path per vault is defined in the *taxis.conf*, through the following entry:

```
<entry key="export_[vaultName]">[path_to_exported_vault]</entry>
```

Moreover the tool offers the following options regarding the file types that will be exported to the exported file structure of the selected vault:

All: all files will be exported.

None: nothing will be exported (disabled).

Specific mimetypes: only files with file extensions that are associated with the selected mimetypes will be exported.

Specific file extensions: only files with file extensions that are manually defined in the “Filters applied” field will be exported.

The selected configuration will be applied to data that will be imported to the system, after its setup.

To apply the current configuration to existing data the **DM Export** web service should be executed.
(http://localhost:8080/spdrm/migration.html)

4.4. External file system

4.4.1. Introduction

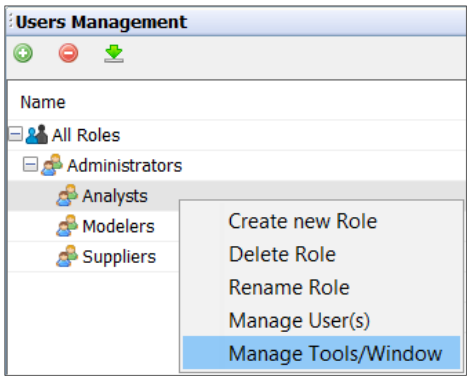
External File System (EFS) functionality gives access to external file systems (additionally to the SPDRM File System) to all SPDRM Users from within the SPDRM Client.

SPDRM Users can navigate through a data tree in external file systems and perform basic actions (view, export and import), even when they don't have access rights on the target file path. Obvious prerequisite is the access rights on those paths of the external file system for the SPDRM Server's user ID.

External File System functionality consists of two modes:

- **External FS Management** mode is accessible only for SPDRM Administrators and is used for the set-up and management of the base paths, relative directory paths (from the base paths) and file extensions, as well as for setting privileges.
- **External FS View** mode is accessible for all users and can be used for the navigation through the external file system and for performing actions. SPDRM users are able to navigate in the data tree and read content in external file systems, no matter if they have access rights on the target file path or not. The entire EFS data is read-only for all users.

In order to give access to External FS View Mode for selected user groups open the **Users Management** window (Resources > Users Management). Access the context menu of the desired group, and activate the option **Manage Tools/Windows**.



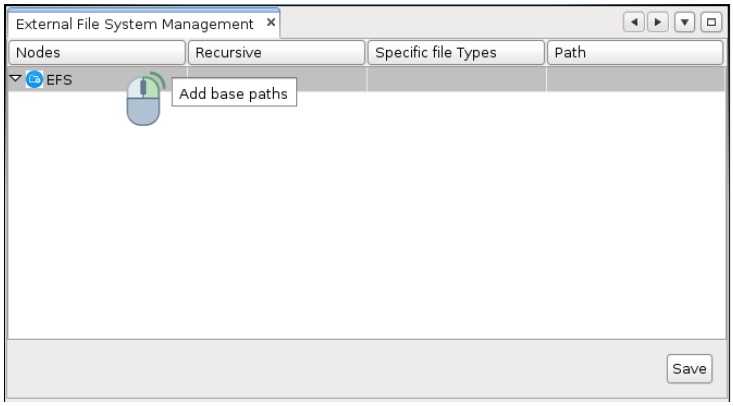
4.4.2. External File System Management

4.4.2.1. Add base path and set ACLs

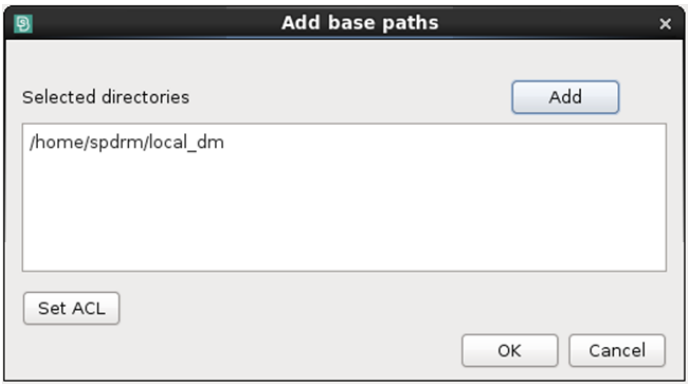
The **External File System Management** window is accessible only to administrators through Data > External File System > External FS Management.

First view of the EFS Management Window contains only the top level EFS Icon.

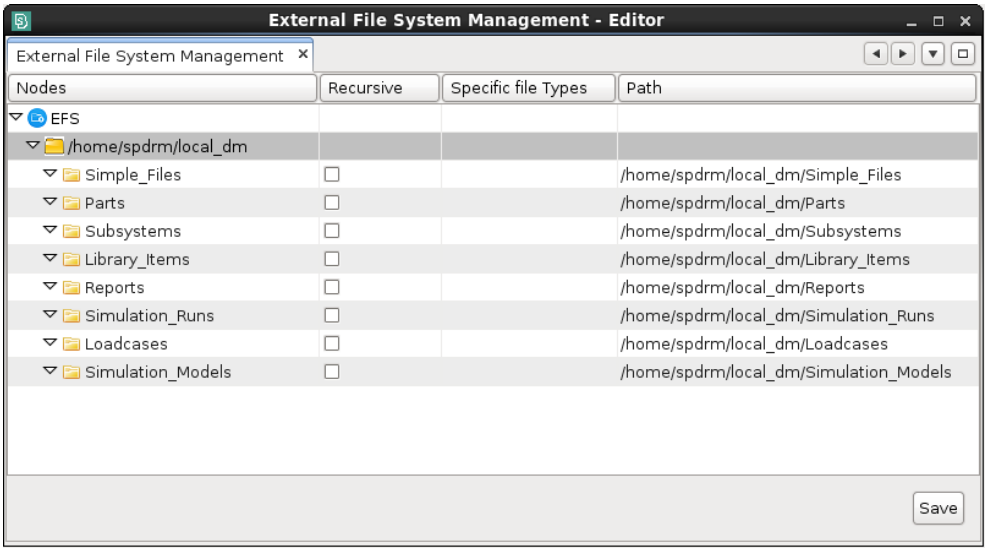
With **right click** on EFS Icon and **Add base paths** option the **Add base paths** window appears. By selecting **Add** on this window, the administrator can select the Base Path directory of a single or multiple EFSs.



On the same window the administrator can set desirable ACLs for all Groups, by selecting **Set ACL**.



After the selection of the Base Paths, the contents of the EFS can be seen in tree structure, along with the following three columns:

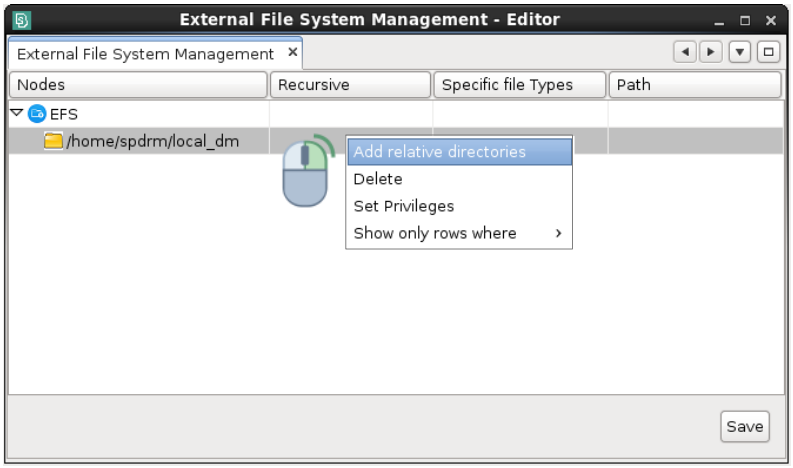


- **Recursive,** which is a value indicating if the selected directory is searched recursively or only at top level.
- **Specific File Types**, which informs about the file types that are searched and displayed in every directory.
- **Path**, which displays the full path of every directory in the EFS.

4.4.2.2. Add and configure relative directories

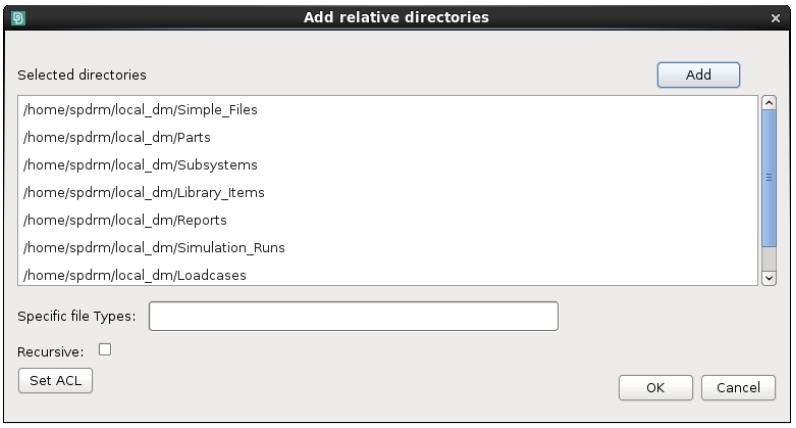
Three actions are available in the context menu of a listed base path:

- **Add relative directories:** Enables the selection of the Relative Directories that will be added in EFS View.
- **Delete:** Allows the deletion of Relative Directories that have been added in previous step.
- **Set Privileges**: Sets the ACLs for the Relative Directories.



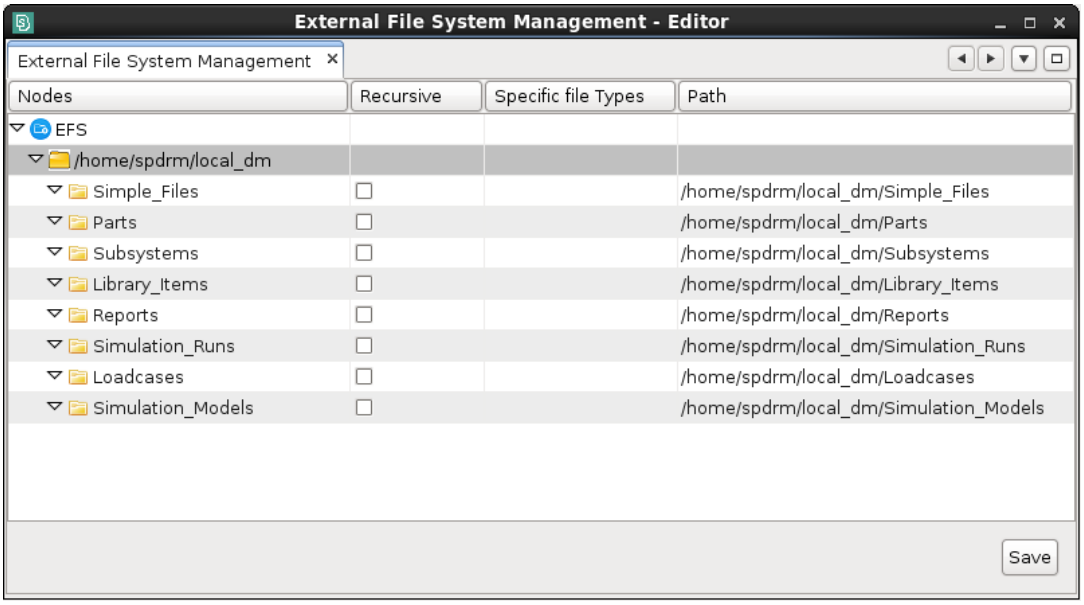
By selecting **Add relative directories**, a window opens for the configuration of the directories. On that window the user must press the **Add** button and then select directories from the *File Manager* that opens.

The Selected Directories are then displayed in the Add Relative directories window.



The Add relative directories window has now 3 configuration fields:

- **Specific file Type:** A text field, where the types of the files in EFS that will be displayed are written (comma separated).
- **Recursive:** A check box, which defines if the children folders of the selected relative directories will be displayed recursively or not.
- **Set ACLs:** A push button that opens the Set ACLs window, as happened in set base path also.

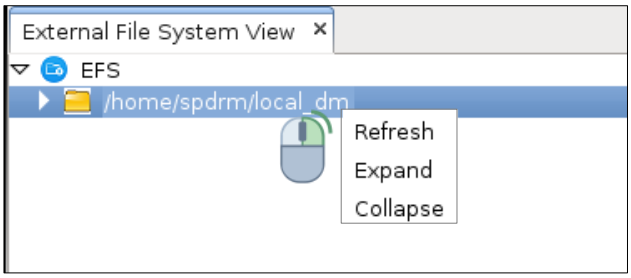


4.4.3. External File System View

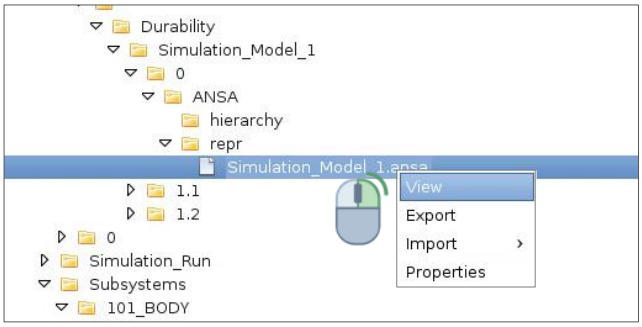
4.4.3.1. Navigate

External File System View window is accessible to all user groups with view privileges from the administrator, through Data > External File System > External FS View.

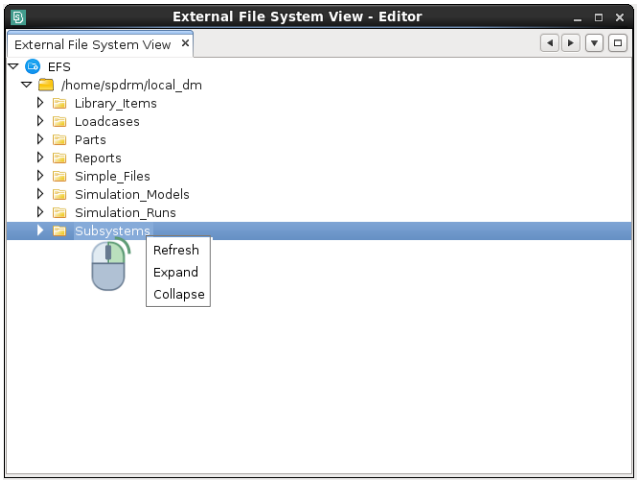
When External File System View window opens, the **Base Paths** and their **Relative Directories** set by the administrator are displayed. By **right-clicking** on a **Relative Directory,** the context menu with three options opens (**Refresh**, **Expand** and **Collapse**).



When **Expanding** the Data Tree, only the files of data type selected in EFS Management are displayed.



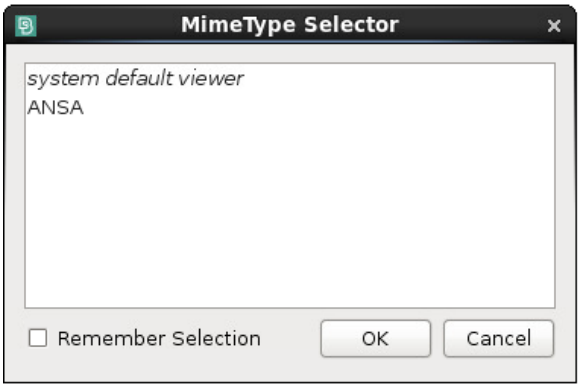
The reverse action happens when **Collapse** is selected. The same context menu appears from every virtual folder in the Data Tree, performing the same actions, starting from the specific structure level.

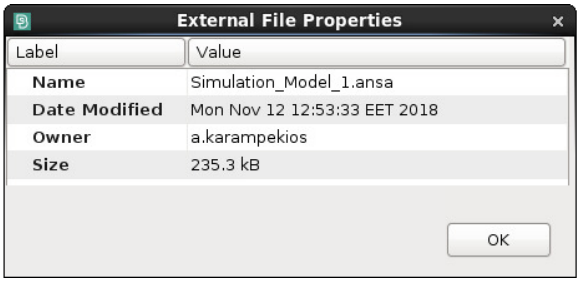


4.4.3.2. View and properties

The following actions are available for each file in the EFS Structure:

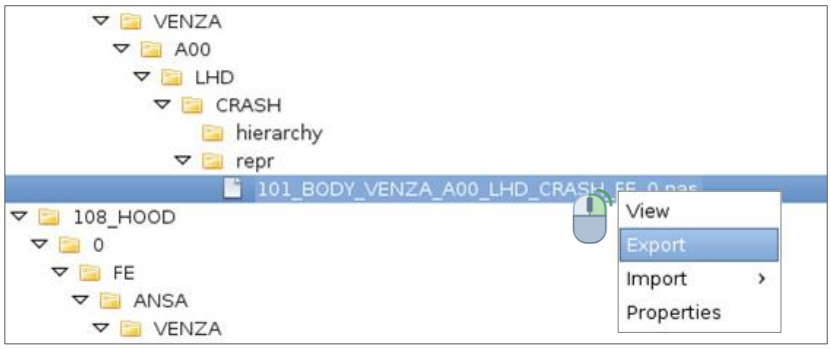
- **View:** Allows viewing of a file using the Registered Applications set in the SPDRM environment. When **View** is selected, the Mime Type Selector appears, asking for the Registered Application to be used for viewing. Notice that the file selected opens in 'View Only' mode, since there are no write privileges in the External File System for the user.
- **Export:** Allows the export of the file in another file-system and is described in the next slide.
- **Import:** Allows the import of the file in the SDPRM database and is described in the next slide.
- **Properties:** Basic properties of the External File are displayed **(Name, Date Modified, Owner, Size). Properties** action is static and no other action is performed.

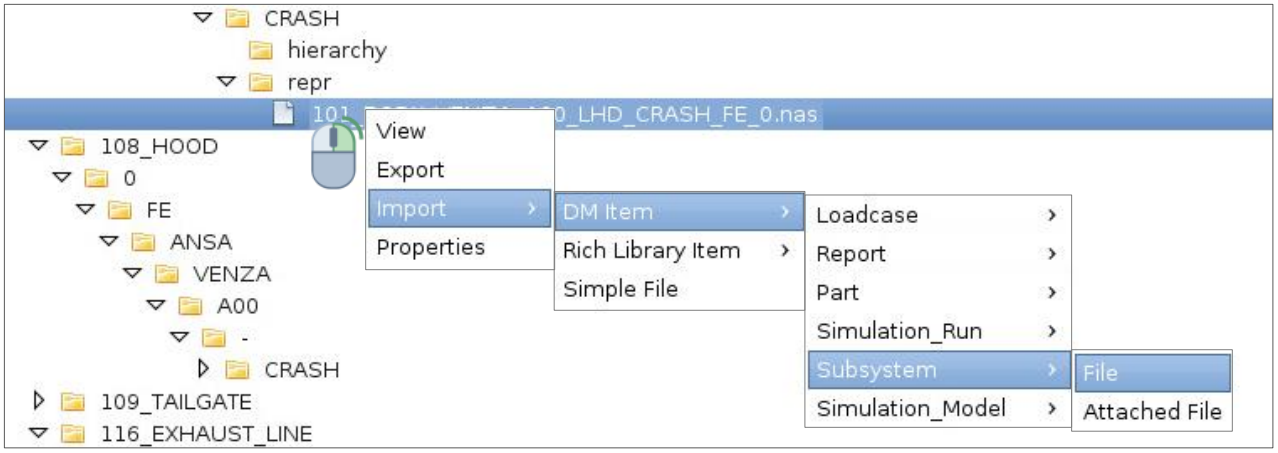




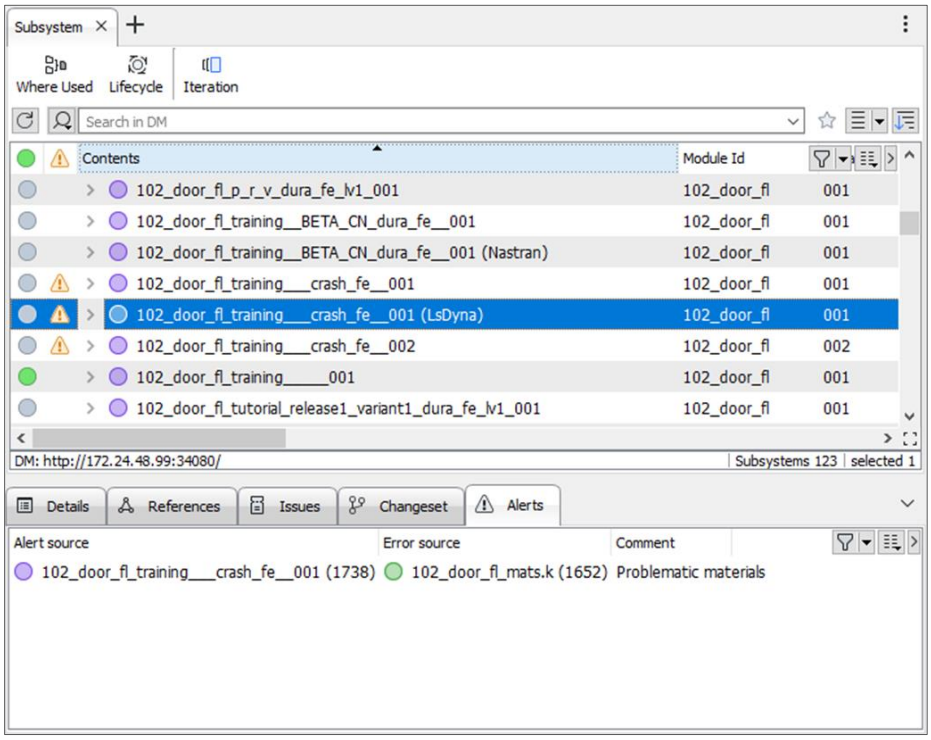
4.4.3.3. Import/Export

The user has the ability to **Export** selected files from the External File System to a File System with full privileges for the user, by selecting the **Export** option from the context menu of the items.





4.5. Alert Mechanism on data



An alert mechanism on data can be set-up in SPDRM. This mechanism can raise alerts for the problematic objects and propagate them to all associated objects, offering the means to a complete error impact analysis.

The alert mechanism can be customized per Data Type through the `dmAlerts.json` configuration file. It is possible to control the Attribute and Attribute Value which will trigger the alert for subsequent data objects, as well as the alert message. An example of the format of the file is given below:

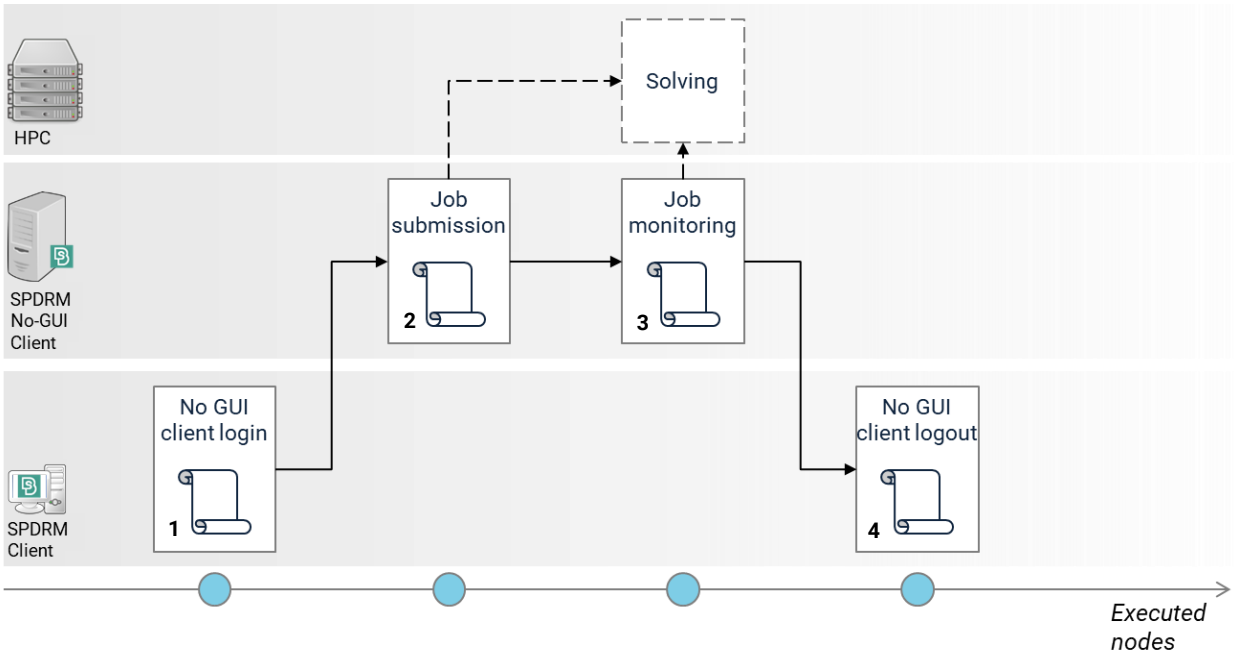
```
"Subsystem": [
    {
      "message_description": "${Comment}",
      "attribute": "Status",
      "alertValues": [
        "Error"
      ]
    }
  ]
```

In this example the Subsystem is configured to take an alert whenever its attribute *Status* takes the value *Error*. Additionally, the *Comment* of the alert will be generated from the *Comment* attribute of the DM object with the Error Status.

Such configuration can be defined in this file, for other DM Objects or/and other Attributes.

5. Process Management

5.1. SPDRM No-GUI Client setup



The assignment and execution of tasks in a No-GUI SPDRM client is possible. This capability can be used for the execution of heavy and time-consuming jobs, on remote resources (even on different sites of an organization), such as the job submission of a Simulation Run to the solver, the polling of the solver results, and the post-processing of the results.

The default setup enables the configuration of the SPDRM No-GUI Client in such a way that it runs on a machine different than the one that hosts the SPDRM application server, to avoid its overload.

This can be done by adding the **noGuiClientSshHost** key in the server's configuration file (*taxis.conf*), e.g.

```
<entry key="noGuiClientSshHost">SPDRM_NOGUI_SERVER.DOMAIN_NAME</entry>
```

NOTE: SSH keys should be created for each SPDRM user, in order to enable the SPDRM Server ID to open SSH connection on the `SPDRM_NOGUI_SERVER` on behalf of the `CLIENT_USER_ID`, without prompt for password.

This means that the SPDRM Server ID should be able to execute successfully the following command:

```
ssh CLIENT_USER_ID@SPDRM_NOGUI_SERVER.DOMAIN_NAME
```

After having established the SSH connection the SPDRM Server will execute the startup script of the No-GUI SPDRM Client, by executing on the `SPDRM_NOGUI_SERVER` the command that is defined in the **noGuiClientPath** key in the *taxis.conf*, e.g.

```
<entry key="noGuiClientPath">/opt/spdrm/client/startClient-nogui.sh</entry>
```

(this process will executed as CLIENT_USER_ID on the SPDRM_NOGUI_SERVER)

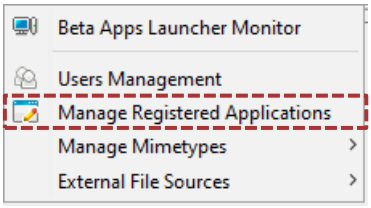
The execution of SPDRM No-GUI Client sessions can be also performed by using the BETA Apps Launcher. Setup instructions of the No-GUI Client with the BETA Apps Launcher can be found in the BETA Apps Launcher paragraph 10.

6. Resources Management

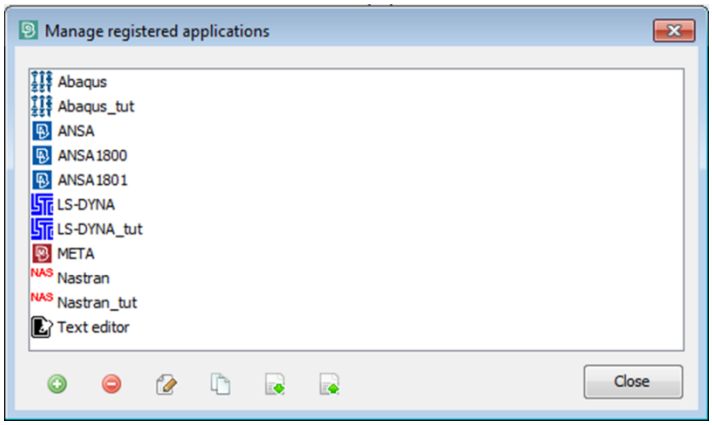
6.1. Registered applications

The user can register applications in SPDRM to be used as file editors or for the execution of tasks in processes. Once registered in the system and their running options are defined, registered applications can be launched directly through the SPDRM interface.

The functionality used to register an application is accessed through the **Resources > Manage registered applications** option from the SPDRM client menu bar.



Selecting the **Manage registered applications** option, the *Manage registered applications* window appears, displaying the list of applications that have already been registered in the system (if any).

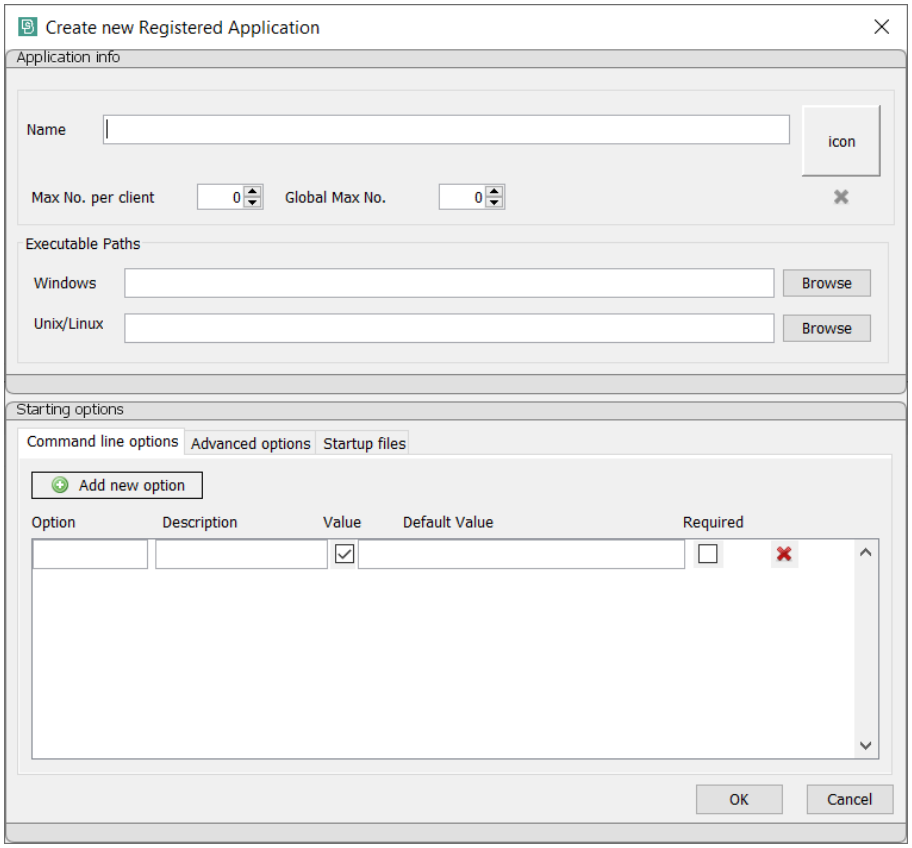


The buttons at the bottom of the *Manage registered applications* window offer various options for the management of registered applications.

- **Create new registered application**: Used to add a new application.
- **Delete registered applications**: Used to delete the selected application.
- **Edit registered application**: Used to edit the settings of the selected application.
- **Create a copy of the registered application**: Used o create a copy of the selected application.
- **Export registered application settings**: Used to massively export the existing application settings.
- **Import registered application settings**: Used to massively register applications by importing a file.

The applications used during the simulation process, as well as the default running options for each application can be registered and controlled centrally on the server side and thus, it is assured that all clients connected to the server always use the same, well-defined, up to date application running options.

Pressing the **Create new registered application** button ( ) the *Create new Registered Application* window appears, where users with the appropriate privileges can define the application information and default running settings of the new registered application.



The user can specify the **Name** of the registered application, as it will appear in the SPDRM client.

The user can select the **Icon** that will appear in the SPDRM client for the registered application.

The user can define the maximum number of the current application instances that can be executed by a client or by an SPDRM environment simultaneously. The first can help avoid overloading of the client workstation, while the second avoid exceeding the license limit of an application.

The path that points to the executable of the application, for Windows and/or Unix systems respectively, can be typed in the respective fields or the administrator can browse through the system File Manager by pressing the **Browse** button in order to access the path.

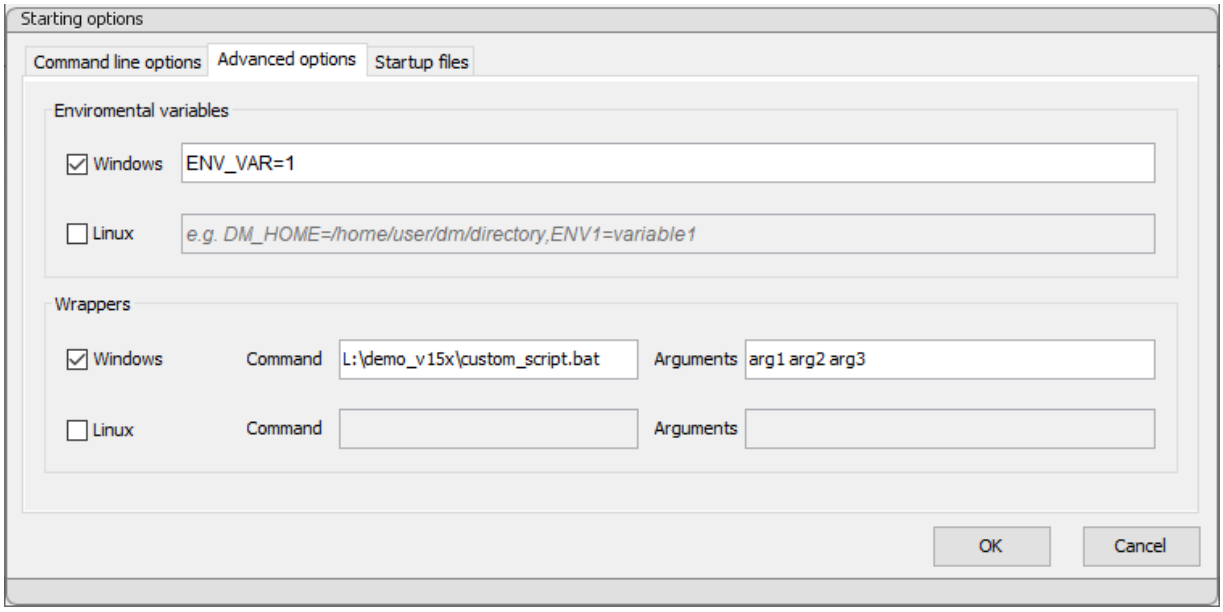
Note that browsing through the File Manager cannot be used for defining the path to an operating system other than the one of current machine, thus it must be typed manually.

One or more starting command line options must be defined in the *Command line options* tab. The options for the application can be set by clicking on the **Add new option** button. Any execution option that the specific application requires can be added.

The **Value** check box next to the description of the execution option, determines whether a value is required after the switch, while the **Required** check box defines whether the specific option is mandatory for the application or not.

In case a default value for a specific option needs to be set, the **Default Value** field must be filled in.

To remove one or more starting options, the user can press the respective button of each option ( ).



In the **Advanced options** tab, additional options can be defined for an application.

Environmental Variables:

Pairs of environmental variables names and values can be defined here, comma separated. Every time this application is called, these environmental variables will be set before its launch.

Wrappers:

A wrapper script can be defined. Every time the application is called, this script will be executed instead. The executable path of the application, and also the options marked as required, are passed as arguments to the custom script. Additional arguments can be passed to the script using the field **Arguments**.

The order of the arguments used when the command will be constructed is:

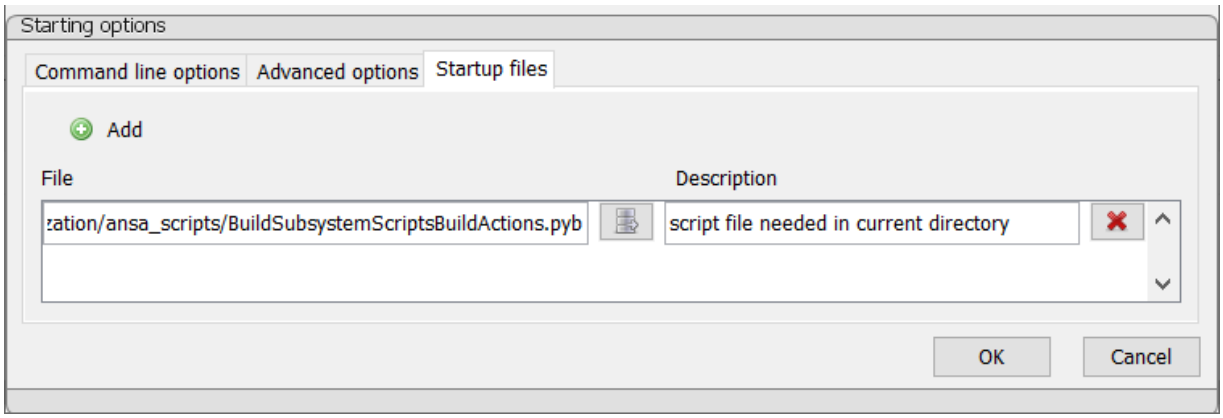
1st: **Arguments** field, if filled

2nd: **Executable path** of application

3rd: File to open, if given

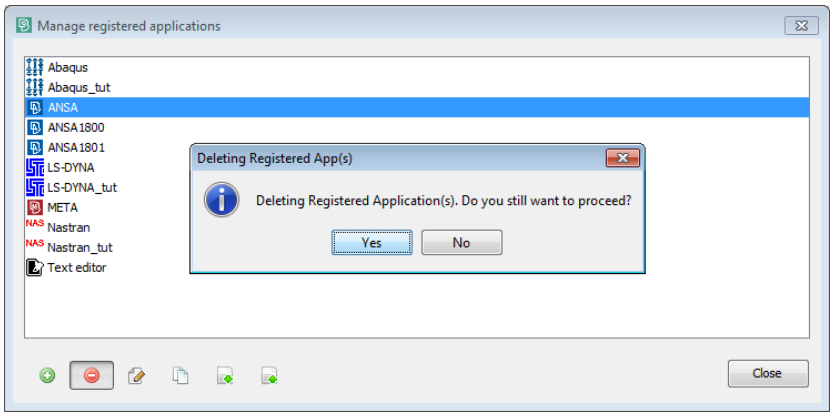
It is implied that the first part of the constructed command is always the custom script path.

Note that the contents of the field **Arguments** are passed intact (e.g., comma separated, with spaces, etc.) and the writer of the wrapper script should take care their processing.



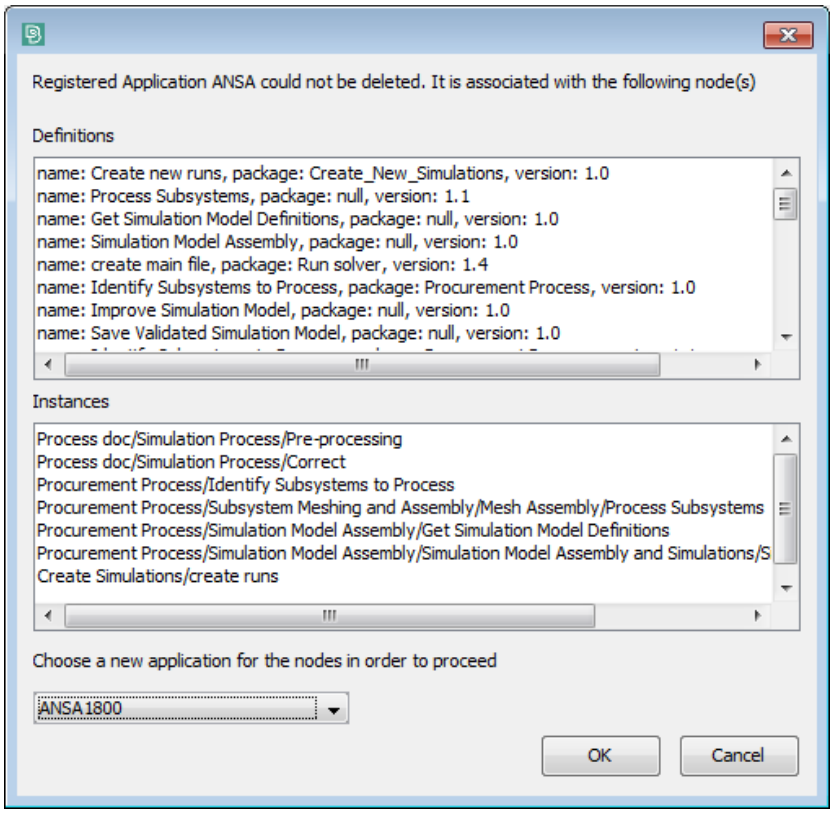
In the **Startup files** tab, auxiliary files from the DM library can be set in order to be downloaded in the execution directory of the application upon launch.

In order to delete one or more existing registered applications, the user should select them in the list displayed in the *Manage registered applications* window and press the **Delete selected registered applications** button ( ).



The *Deleting Registered App(s)* window appears where the user needs to confirm whether to proceed with the deletion of the selected applications or not.

If the user selects to proceed with the deletion of a Registered Application that is used by one or more process nodes, the application will not be deleted unless the user chooses another registered application for the nodes.

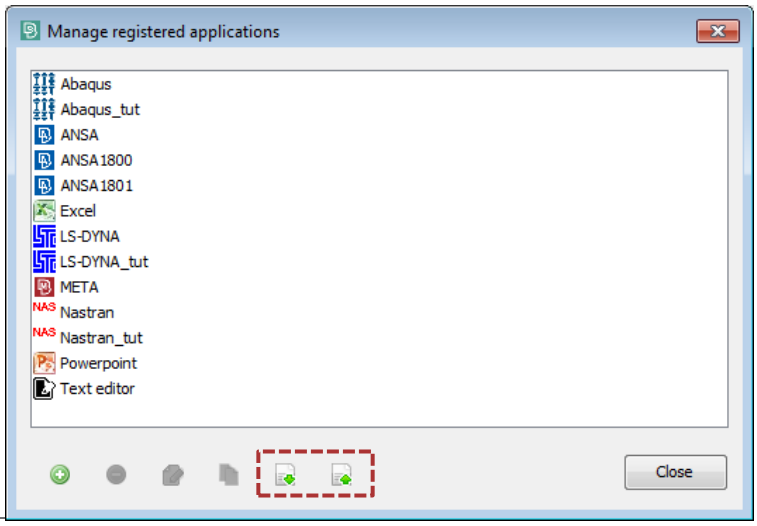


A new window appears where all the Definition Nodes as well as Nodes used during process execution as instances by one or more users are displayed in the *Definition* and *Instances* sections, respectively.

The user can select from a drop down list containing all registered applications, the new application that will replace the current Registered Application, for all the Nodes that use it.

Finally, by pressing the **OK** button, the registered application will be deleted.

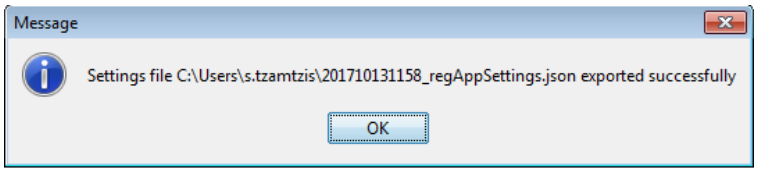
The user can massively **Export** ( ) or **Import** ( ) **registered application settings** using the respective buttons.



Pressing on the **Export registered application settings** button, the user has to select a directory on the local File System where the existing registered application settings will be exported as a *.json file.

Once the directory where the registered application settings will be exported has been

selected, SPDRM exports the *.json file and a Confirmation Message pops-up.



Pressing on the **Import** button, the user can select a *.json file with the registered application settings that will be imported.

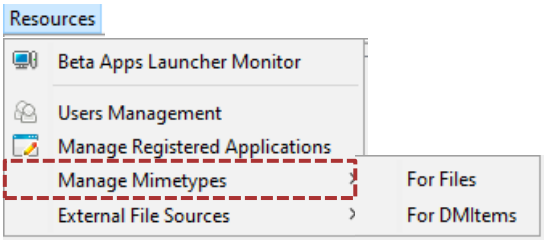
If the file being imported contains registered applications with the same name as an already existing registered application, the user can select to:

1. **Yes**, overwrite the existing application
2. **No**, do not overwrite the existing application and import a new one, adding the suffix *imported_copy*.
3. **Skip**, skip the import of the current RegApp and continue on the next one.
4. **Abort**, cancel the import procedure entirely

Apply to All , will apply the selected action to all the incoming Registered Applications.

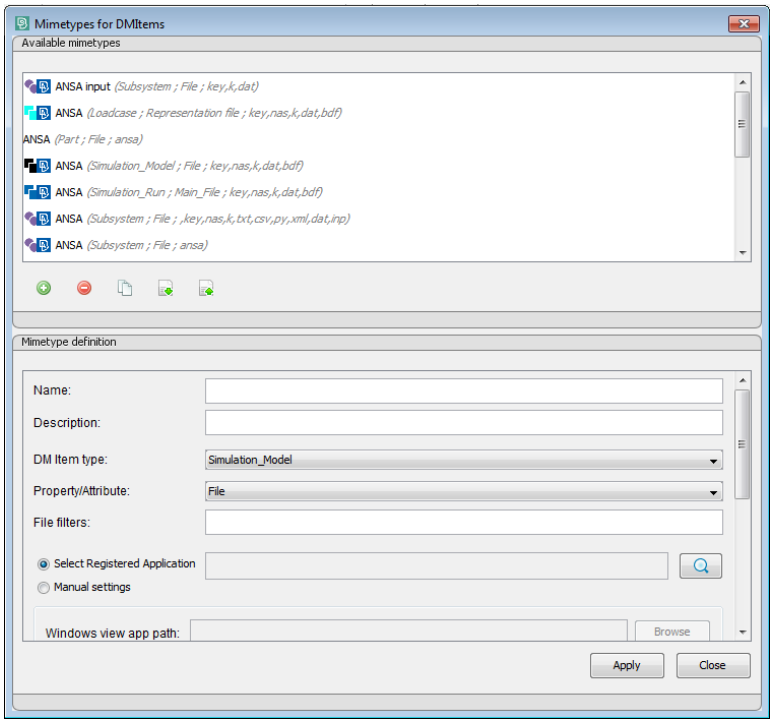
6.2. Mimetypes

Direct view of the data is enabled through the SPDRM client by associating appropriate applications to specific file-types or DM items. Mimetypes control the alternative applications that will be suggested for opening a file on "Edit".



The functionality used to manage mimetypes is accessed through the **Resources > Manage Mimetypes** option from the SPDRM client menu bar.

Selecting the **Manage Mimetypes > For files** or **For DMItems** option, the *Mimetypes* or *Mimetypes for DMItems* window, respectively, appears on screen.



The *Available mimetypes* section displays the list of mimetypes that have already been defined (if any), and the *Mimetype definition* section displays the settings for their definition.

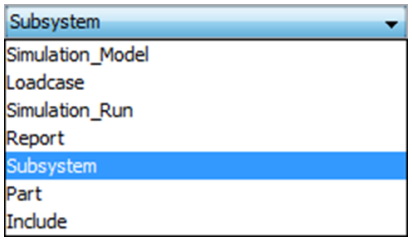
NOTE: System default viewers are automatically recognized as mimetypes from the software. Thus, file types associated with a particular editor on the OS do not need extra care in order to be viewed in SPDRM.

Pressing the **Create new mimetype** button ( ) the user can define in the *Mimetype definition* section the new mimetype settings.

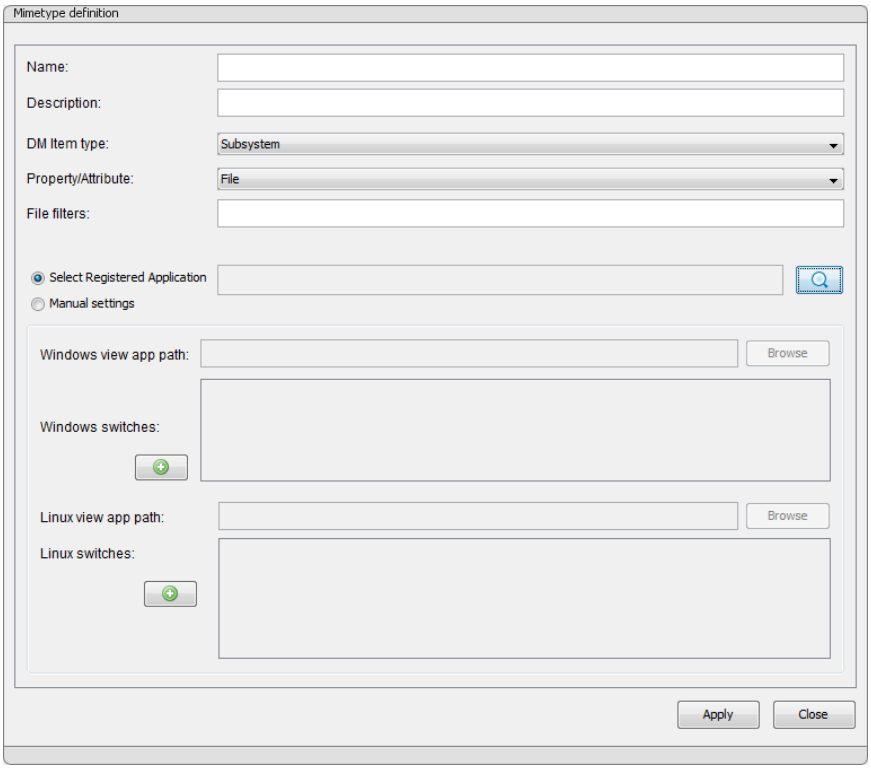
The **Name** of the mimetype, as it will appear in SPDRM client, is defined in the respective field.

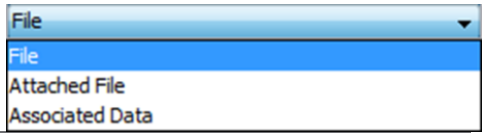
A **Description** can optionally be defined.

The **DM Item type** for which the new mimetype is added can be selected from a drop-down list.



The **Property/Attribute** that contains the file that will be viewed using the new mimetype can be selected from a drop-down list, e.g for a Simulation Run, it would be possible to define a

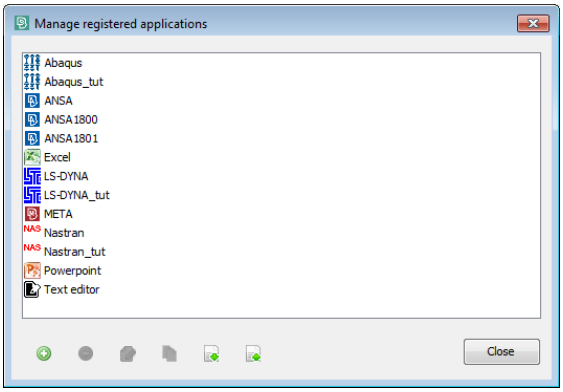




mimetype to edit its keyword file in a text editor and another mimetype to edit its definition file in ANSA. The Properties/Attributes listed depend on the selected DM Item type.

Note that the **DM Item type** and **Property/Attribute** fields only appear when mimetypes for DM Items are being managed.

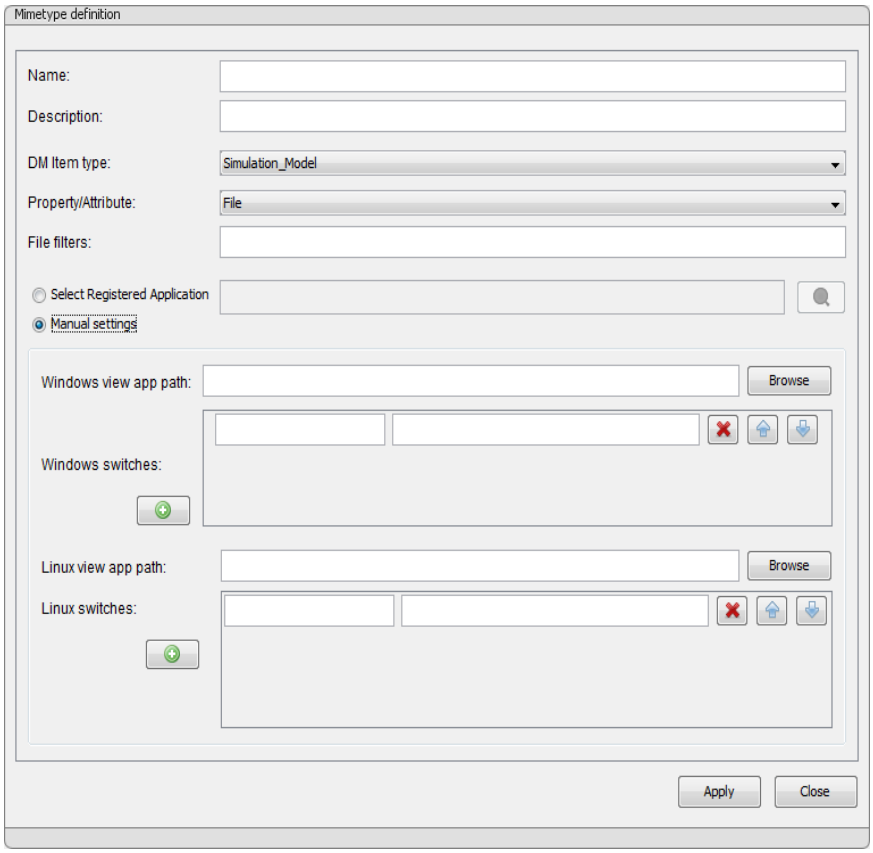
The file types that can be viewed with the new mimetype must be typed in the **File filters** field. If more than one type of files are to be viewed with the particular mimetype, then all the respective file extensions should be typed, separated with commas with no spaces between them.



An existing registered application can be set as a mimetype by pressing the respective button ( ). In this case, all the required arguments are automatically transferred on the mimetype definition settings as well. The user can set this by pressing on the respective button and select one of the existing registered applications.

When updating a registered application, all related mimetypes are getting automatically updated as well.

If the user wants to use as mimetype an application that has not been registered in the system, then the mimetype settings need to be defined manually.



The app path for Windows and/or Linux should be typed in the respective fields, or the user can press the Browse button to browse through the File Manager and access the path.

Note that browsing by using the File Manager cannot be used for defining the path of an operating system different than the one of the current machine thus it must be typed manually.

Command line switches for Windows and/or Linux can be optionally defined.

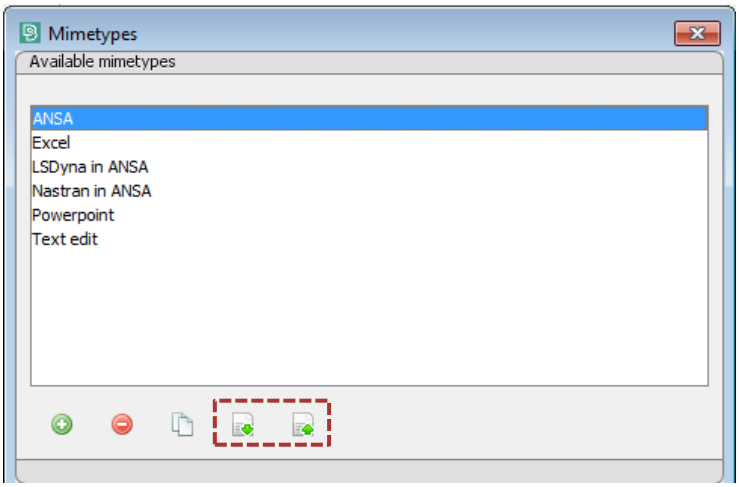
By pressing the **New additional switch** button ( ) the user can add additional switches that may be used in order to view a specific file type.

To remove a switch, the user can press on the respective button ( )

Finally, pressing on the **Apply** button, the definition of the mimetype is finalized and it can be used for viewing the selected files.

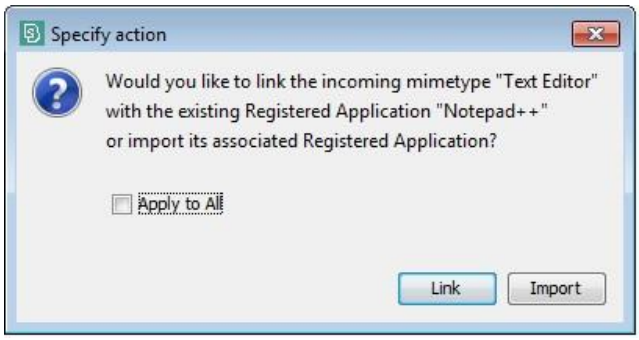
The user can massively **Export** ( ) or **Import** ( ) **mimetype settings** using the respective buttons.

Pressing on the **Export mimetype settings** button, the user has to select a directory on the local File System where the existing mimetype settings will be exported as a *.json file. Once the directory where the mimetype settings will be exported has been selected, SPDRM exports the *.json file and a Confirmation Message pops-up.

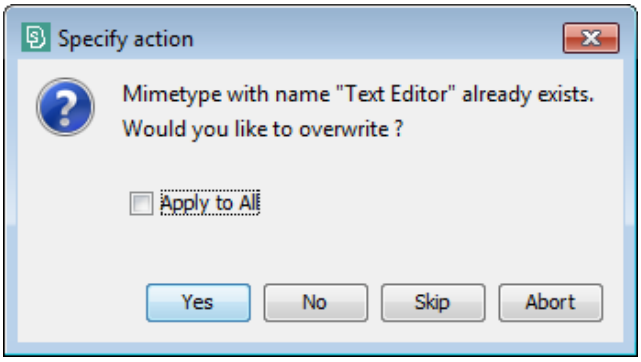


Pressing on the **Import** button, the user can select a *.json file with the mimetype settings that will be imported.

If the incoming mimeype is linked to a Registered Application and a conflict exists between the incoming associated Registered Application and an existing one, the user can select to **Link** the incoming mimetype with the existing RegApp or **Import** it's associated RegApp by renaming it, adding the suffix **imported_copy** to it.



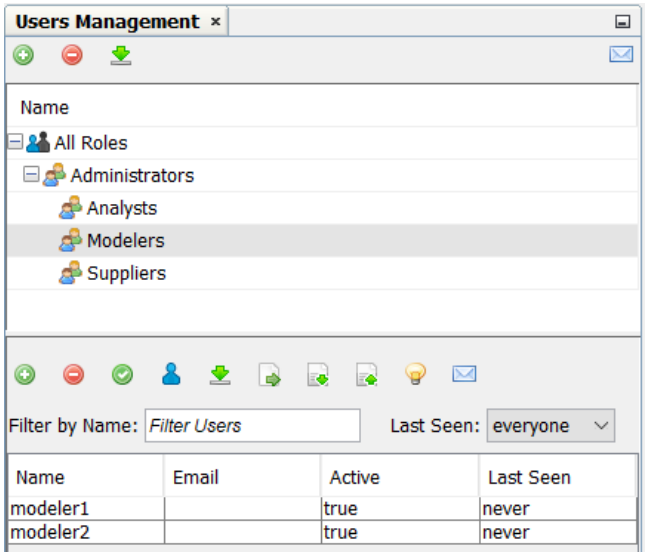
If the file being imported contains mimetypes with the same name as an already existing mimetype, the user can select whether to overwrite the existing mimetype settings or not (there is also the option to apply the same action on all conflicts detected).



6.3. Users management

SPDRM supports the definition and management of different Roles (Groups) each of which contains different users with privileges regarding the SPDRM functionalities corresponding to the role.

Roles and Users can be defined and managed by the Users Management window, invoked through *Resources > Users Management*. This window is only accessible by Administrators and all actions described in the following sections – unless clearly stated – can only be performed by users which have administrative privileges.

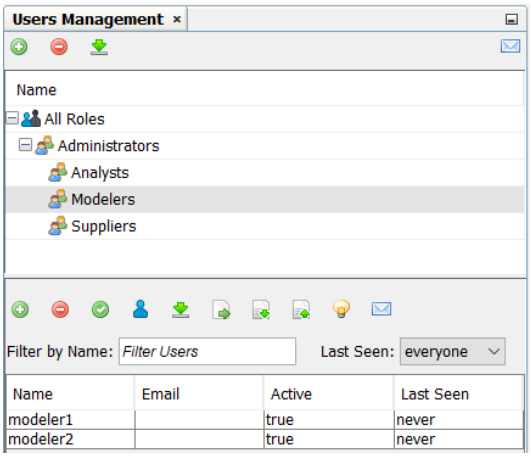


An Administrator can use that window to perform the following actions regarding the SPDRM Users:

- Create and Manage Roles
- Fetch Groups and Roles
- Create and Manage users
- Activate users
- Deactivate users
- Import users
- Export users
- Fetch Users
- Show Logged-in users
- Send e-mail notifications

6.3.1. Create and manage roles

Users Management window is separated in two fields, as can be seen on the image below:



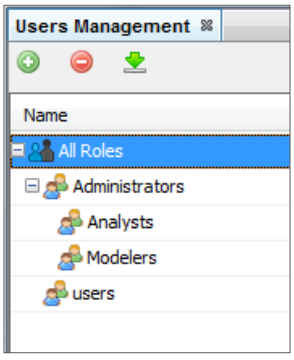
1. **Roles** field
2. **Users** field

Every role is a group of users that defines the data privileges of the users, as well as the access that these users have to the tools and windows of the SPDRM Environment. Every user must belong to at least one role.

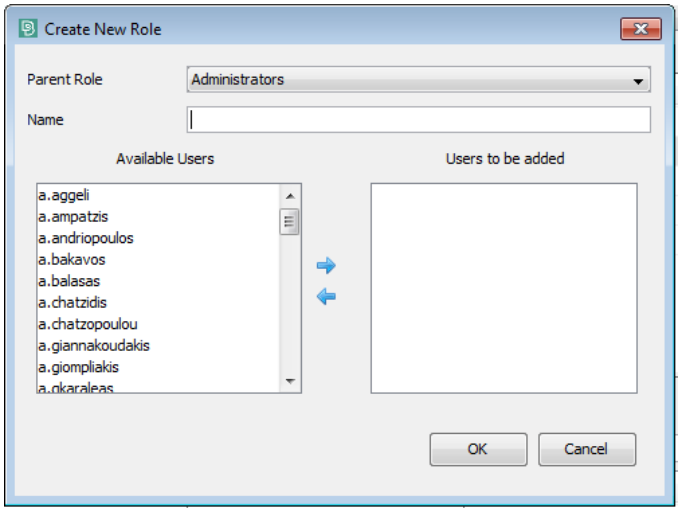
The existence of roles is highly important for the following reasons:

- Roles creation supports a parent-child relationship. This can be seen in the next example, where Modelers, Analysts and Suppliers roles belong to the parent Administrators role.
- The hierarchy of roles is used for the propagation of privileges to higher level roles.
- Users can be added to one, or more roles. This makes it possible for the same user to have different access rights when he/she logs into the system with a different role.
- Role-based access control on SPDRM Client modules (Tools and Windows)
- Default privileges, or privileges of a particular role can be granted on existing data for the newly created role.

All roles are by default displayed when the Users Management window is invoked.



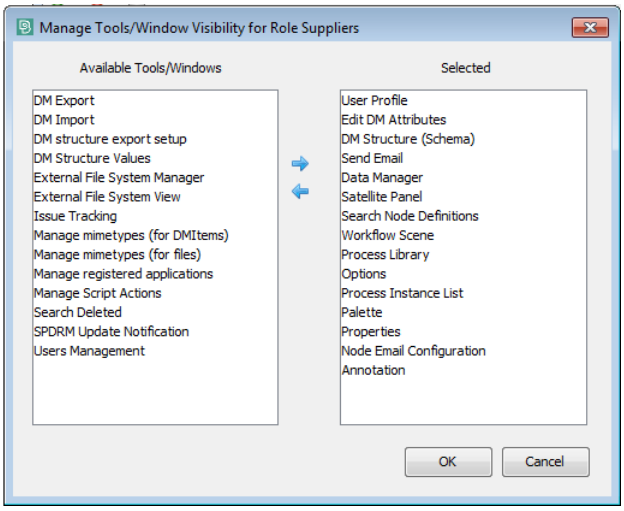
In order to create a new Role, press the  button from the Roles management toolbar at the top of the Users Management window or, alternatively, right click on the **All Roles** items and select the **Create New User** option.



The **Create New Role** window appears. The ability to decide upon the Parent Role (Group) of the newly created group is provided in the **Parent Role** field. In the case of a sub-role creation, one of the available Roles should be selected from the drop down menu. Otherwise, the new role is going to be created as a main group and the option **None** should be selected.

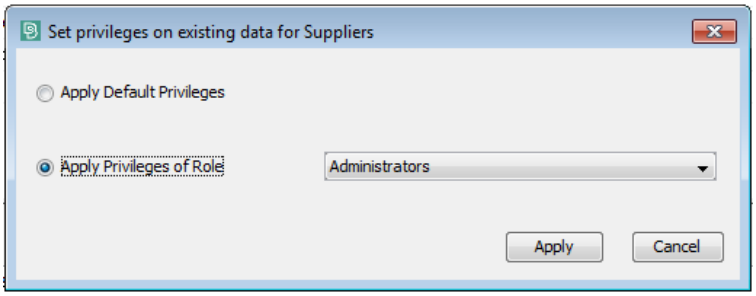
After assigning a name to the New Role, the User comprising the group must be added. One or more Users can be selected from the **Available Users** list on the left and be added to the **Users to be added** list on the right, by using the  icon. The  icon can be used to transfer users back to the left list again.

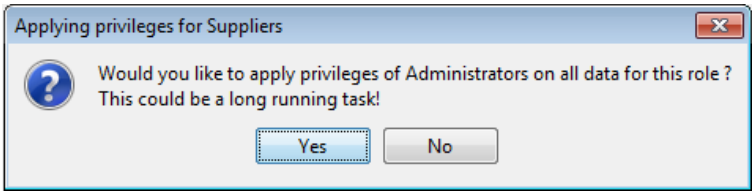
After the users selection, the **Manage Tools/Window Visibility for Role** Suppliers window appears, where the administrator must select which Tools and Windows will be available for the current Role. Tools and Windows can be transferred from the left list with the **Available Tools/Windows** to the **Selected Tools/Windows** to the right and vice versa, by using the  and  icons respectively. In this way it is possible to set role-based access control on SPDRM Client modules (Windows/Tools).



After the selection of the Tools and Windows that will be available for the users of new role, the **Set privileges on existing data for** Suppliers window opens, where the Administrator has two options:

1. **Apply Default Privileges**, as they have been configured in the taxis.conf configuration file
2. **Apply Privileges of Role**, that will use the privileges used by an already existing role to define the privileges of the new role

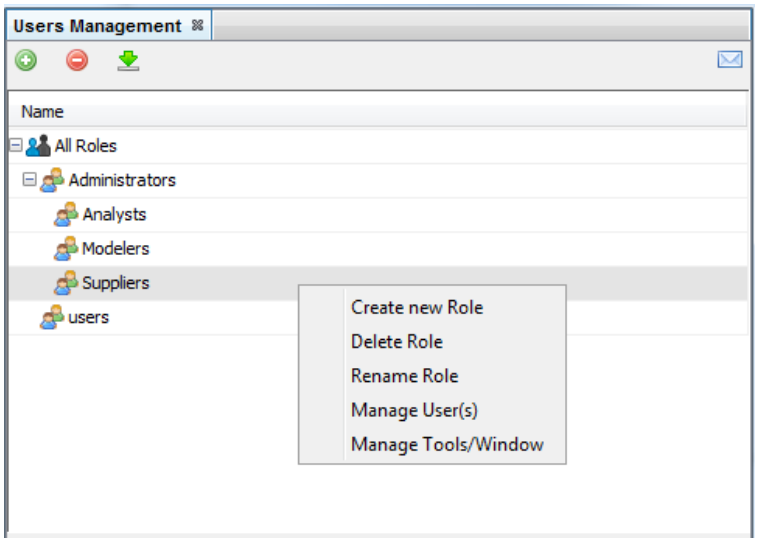




The administrator is asked to confirm the changes and the new sub-role Suppliers is created under Administrators role.

The administrator can perform modifications and other actions to the existing roles at any time, by the using the actions provided on the context menu. The context menu can be accessed by right-click on any role or sub-role and the actions that can be performed are the following:

1. **Create new Role,** that has been already mentioned. **Delete Role**, that deleted an existing role
2. **Rename Role**, that can be used for renaming an existing role name
3. **Manage User(s),** that opens a window similar to window of previous slide and enables the addition and deletion of users from the role
4. **Manage Tools/Window,** that opens a window similar to window of previous slide and enables the addition and deletion of Tools and Windows that will

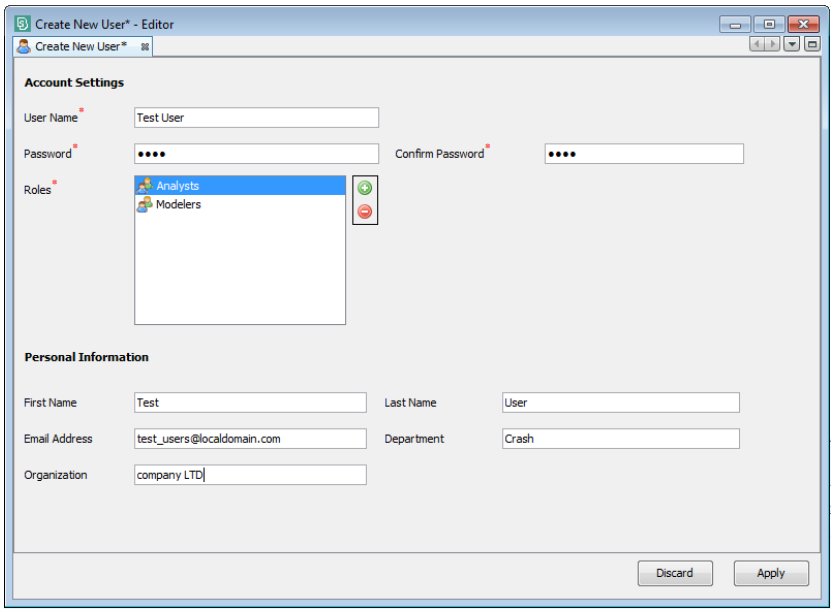


be visible to the users of the role.

NOTE: A 6th option could appear in the context menu of each role that is called **Set Target Vault**. That option opens a new window, where the administrator can define the vault in which the data created by each role will be saved, if more than one vaults are defined. That option appears only when the key **set_target_vault_per_role** is set to **True** in the server's configuration file.

6.3.2. Create and manage users

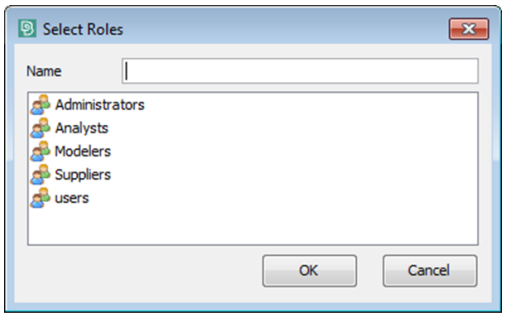
By clicking the button the **Create New User – Editor** window opens where the **Account Settings** and **Personal Information** of a user can be defined. Account Settings must be defined obligatory, while Personal Information field is optional. The button in Roles field opens the **Select Roles** window where the administrator must select in which Roles the new user will belong (more than one can be selected).

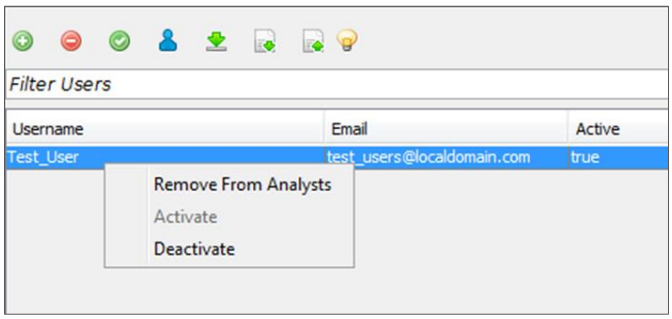


After the creation of the new role, the user appears in the **User field** of each selected role, and from it's context menu three management operations can be done:

1. **Remove** user from Role.
2. **Activate** user (if deactivated).
3. **Deactivate** (if active).

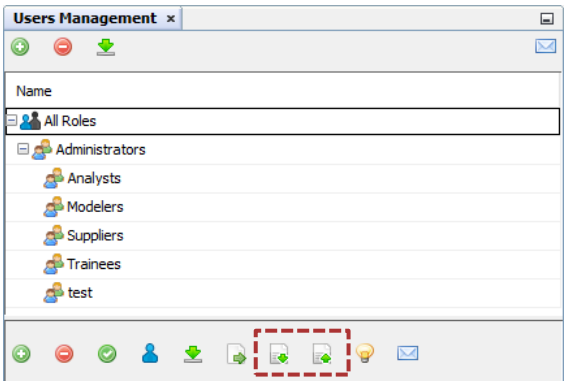
Last two actions, Activate and Deactivate, can be done, also, by using and icon respectively.





6.3.3. Import/Export Users

An SPDRM Administrator can easily export all the information regarding the Roles and Users of an SPDRM environment by using the **Export** button. This button opens a file manager asking for a folder where the .json file with all Roles/Users information that is produced will be exported. That .json file contains all the necessary information regarding:

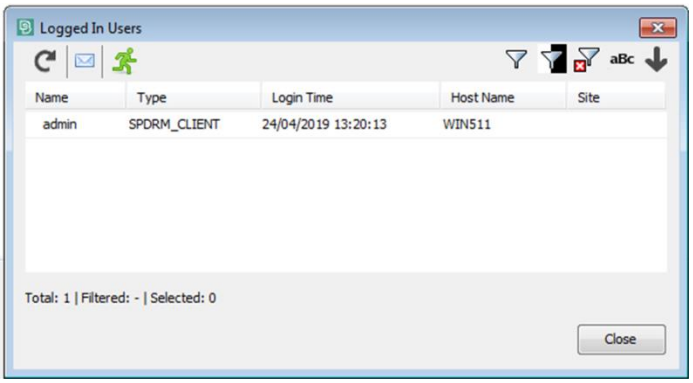


- Roles and users with their respective privileges regarding Data, Tools and Windows.
- Roles' structure (parent – child).
- Active and Deactivated users

By using the **Import** button, the administrator can select a .json file exported from another SPDRM environment, to be imported in the current environment.

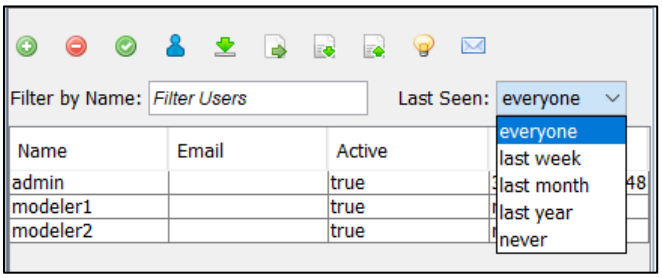
6.3.4. Show Logged-in users

By pressing the **Show Logged in Users** button, the Logged In Users window appears, which displays info regarding the logged in SPDRM users. That tool is mainly used to inform the administrator about the logged in users, when an update is going to take place. Three actions are available from this window:



- **Refresh**
- **Send email to all/selected logged-in users**, that opens the Send Email tool.
- **Close Selected User Sessions**, that automatically closes all SPDRM clients, apart from the client of the administrator.

By using the **Filter by Name** option, a specific user can be searched and by using the **Last Seen** drop-down menu, as shown in, the filtering of the users that have logged-in during a specific time period is enabled.



The last functionality can filter the users that have logged in during **last week**, **last month**, **last year** or **never**.

The filtered users on Users Management window can be selected and by clicking the e-mail icon window , directly send an e-mail only to them.

6.3.5. LDAP authentication

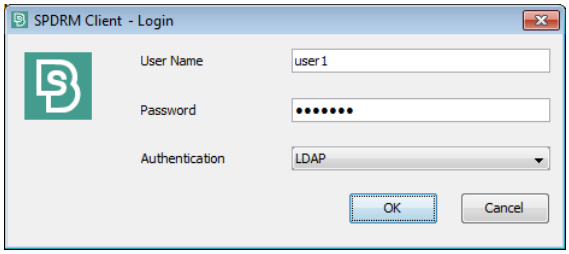
By default SPDRM supports the authentication using SPDRM credentials (username and password). However, SPDRM offers the option to use authentication through LDAP (or Active Directory).

To enable LDAP authentication the LDAP Server URL and the User DN should be set properly in the server's configuration file (*taxis.conf*), e.g.

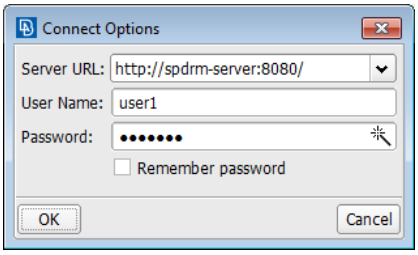
```
<entry key="LDAP_URL">ldap://ldap.server:389</entry>
<entry key="LDAP_username">uid={0},ou=users,dc=domain,dc=net</entry>
```

NOTE: `{0}` is automatically replaced by the SPDRM User Name during login through LDAP authentication

After setting up the LDAP settings the users are able to login the SPDRM Client using the LDAP authentication method.



The LDAP authentication on the SPDRM Server is also supported through ANSA / META sessions.



To force the SPDRM Client to enable only LDAP authentication, the following entry should be added in *taxisprops.xml*.

```
<entry key="default_auth_type">LDAP</entry>
```

To force the SPDRM Server to accept only LDAP authentication from SPDRM Client and/or ANSA/META sessions, the following entries should be added in *taxis.conf*.

```
<entry key="spdrm_auth_type">LDAP</entry>
<entry key="ansa_auth_type">LDAP</entry>
```

6.3.6. Fetch LDAP users in SPDRM

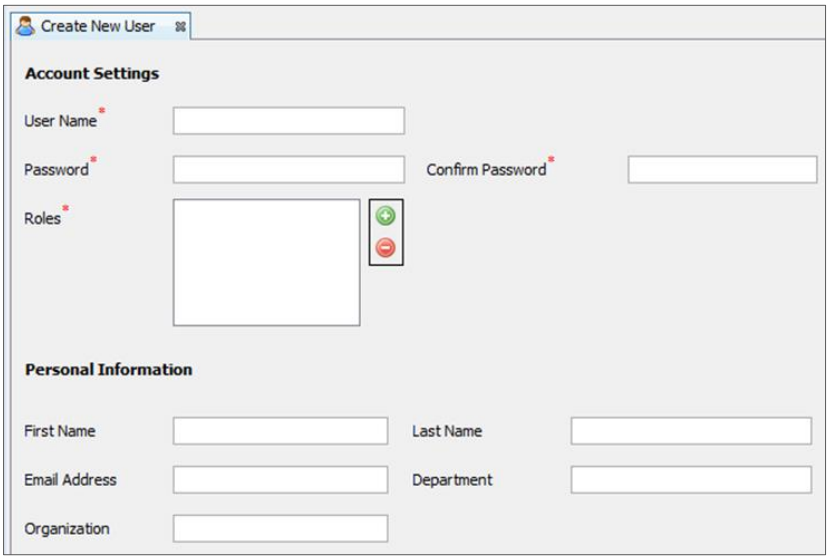
The first step is to configure the SPDRM Client to enable authentication on the LDAP server. To enable LDAP authentication, the following entry should be added in the SPDRM Client configuration files (i.e. taxisprops.xml.windows/.linux)

```
<entry key="ldapMapping">{FirstName:givenName,LastName:sn,UserName:uid,Email:mail}</entry>
```

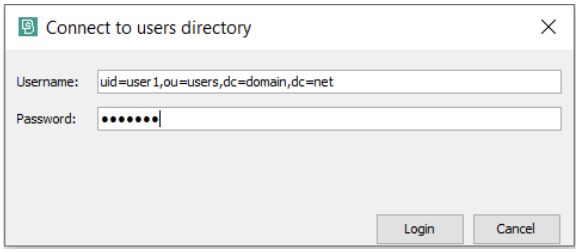
NOTE: To properly define the value of the ldapMapping key, the name of the following LDAP attributes is required, for the fields of the SPDRM User Profile, i.e.:

- User Name (e.g. uid)
- First Name (e.g. givenName)
- Last Name (e.g. sn)
- Email Address (e.g. mail)
- Department (e.g. department)
- Organization (e.g. organization)

The name of LDAP attributes (uid, sn, etc.) may vary from the ones used in the above example. In this case, these should be replaced accordingly in the ldapMapping value.



To connect to LDAP server start the SPDRM Client and login as “admin” (User Name: admin, Password: admin)

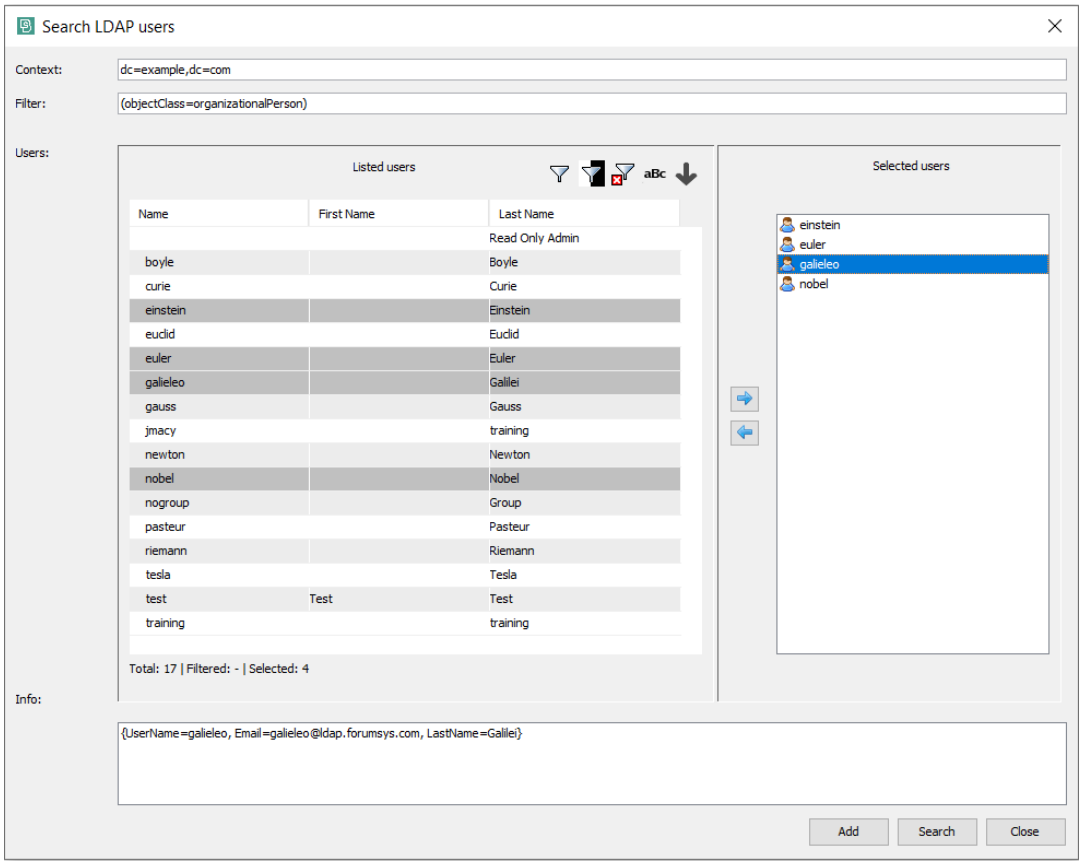


- Go to: Resources > Users Management
- Press **the Fetch Users (AD/LDAP)** button 
- Enter the following information in the pop-up dialog:
  - Username: uid=<YOUR_LDAP_USER_DN>
  - Password: <YOUR_LDAP_PASSWORD>

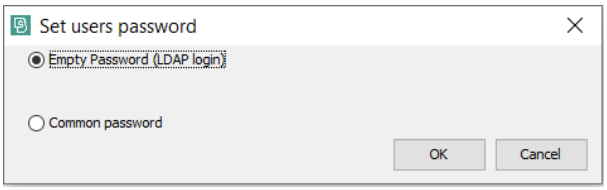
To search LDAP users enter the following information in the pop-up dialog:

- Context: ou=users,dc=domain,dc=net (Base DN)
- Filter: objectClass=organizationalPerson (LDAP objectClass)

Press the **Search** button to get the list of Users.



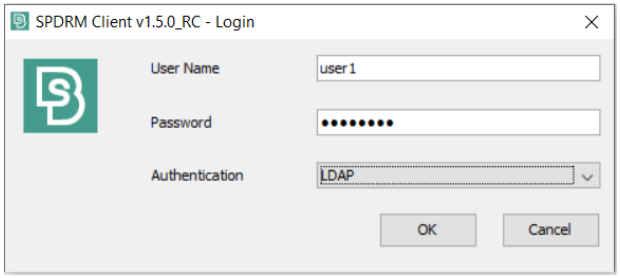
To add LDAP users select from the Users list the ones you would like to add to SPDRM and press the **Right Arrow** button. Use the **Left Arrow** to revert the action. Press the **Add** button and all users in the **Selected users** list will be added. Select the option **Empty Password (LDAP login)** and press the **OK** button.



To assign SPDRM role(s) to imported LDAP users from **Resources > Users Management**, press the **Show All Users** button and select a user from the users list. In the **Edit User** tab, press the **+** button, select the role(s) to be assigned to this user, and press the **OK** button. Press the **Apply** button to save the changes on the user's profile.

To login to the SPDRM Client through LDAP authentication, start the SPDRM Client and enter the following information in the login dialog:

- User Name: <YOUR_LDAP_USERNAME>
- Password: <YOUR_LDAP_PASSWORD>
- Authentication: LDAP



6.3.7. LDAP authentication over SSL (LDAPS)

LDAP authentication over SSL (LDAPS) is also supported by SPDRM, but it requires the following additional configuration steps.

The CA certificate of the LDAP server needs to be installed on the SPDRM Server by following the next steps:

- Open a terminal and change directory to the SPDRM Server's configuration directory: `[SPDRM_SERVER_DIR]/wildfly/standalone/configuration/`
- Generate (or request from the IT department) the SSL certificate file (e.g. *certificate.crt*) of the LDAP server, and copy it into the above directory
- Run the following command to import the certificate file to the keystore: `[SPDRM_SERVER_DIR]/java/jre_linux/bin/keytool -importcert -trustcacerts -file certificate.crt -keystore publicKey.store`
    - keystore password: *changeit*
    - Trust this certificate?: *yes*

NOTE: The above command will generate the keystore file *publicKey.store* into the SPDRM Server's configuration directory.

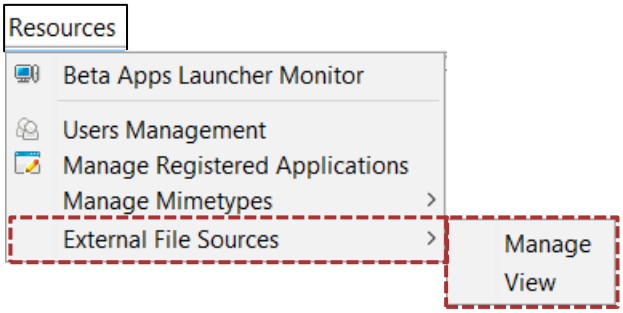
Edit the WildFly application server's configuration file (i.e. *standalone-full.xml*) and add the following properties, just under the existing key *spdrm.conf.location* (i.e. line 35):

```
<property name="javax.net.ssl.trustStore" value="${jboss.server.config.dir}/publicKey.store"/>
<property name="javax.net.ssl.trustStorePassword" value="changeit"/>
```

In case of LDAPS, the LDAP URL entry in the *taxis.conf* should have a syntax similar to the following example:

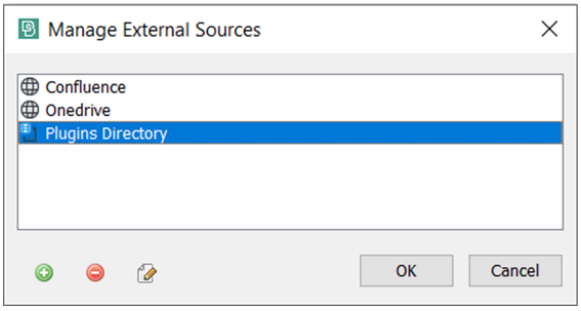
```
<entry key="LDAP_URL">ldaps://ldaps.server:636</entry>
```

6.3.8. External File Sources



External File Sources enable the addition of external files in the Data Tree in terms of URL or mounted network share. The respective tool enables the definition of those sources, as well as their access management. The functionality is accessible through the path **Resources -> External File Sources** and contains two sub-menus, **Manage and View.**

By selecting the Manage menu, the definition of the External Source can take place. The **Manage External Sources** window provides three options:

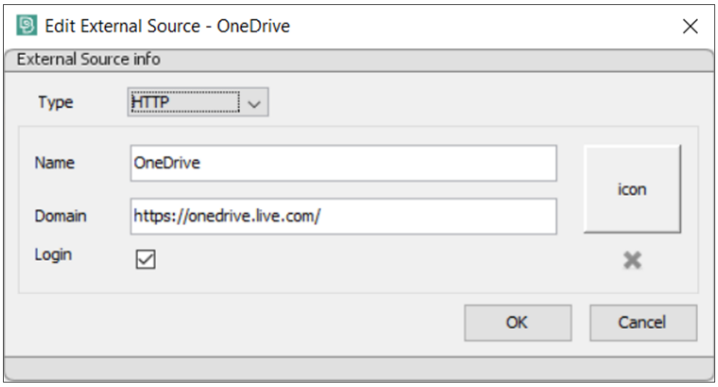


 Create new external source

 Delete selected external source

 Edit selected external source

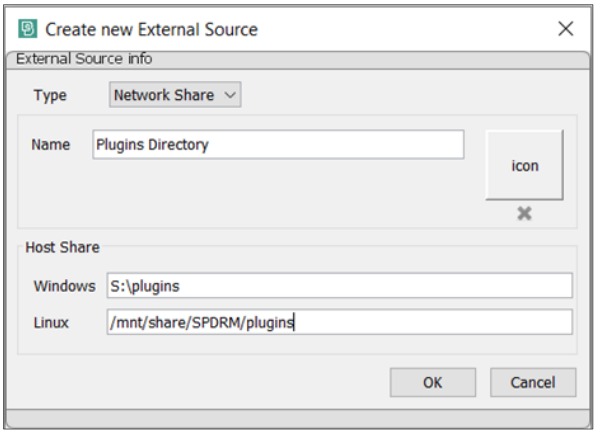
By selecting the Create  button, the **Edit External Sources** window appears where an external source is defined. The type of the External Source must be firstly defined. **HTTP**  and **Network Share**  are the available types. HTTP source will be used for URLs, while Network Share for mounted drives. An HTTP source needs the following configuration:



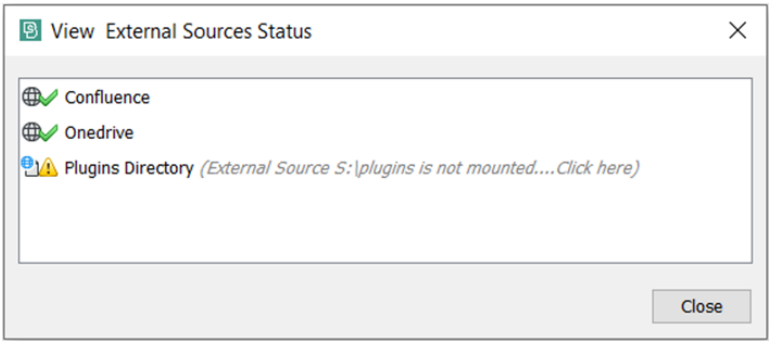
- **Name**: The name of the External Source.
- **Domain**: The domain that hosts the external files.
- **Login**: Checkbox defines if Login to the domain is required.
- **Icon**: A custom icon can be selected from file system.

On a similar window a new Network Share source needs the following configuration:

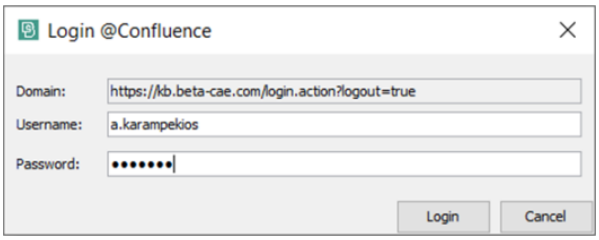
- **Name**: The name of the External Source.
- **Windows Host**: The domain that hosts the external files.
- **Linux Host**: Checkbox defines if Login to the domain is required.
- **Icon**: A custom icon can be selected from file system.



After the External Sources set-up, the user can see all listed External Sources from the **View** menu. External sources with unverified credentials will be shown on this window.

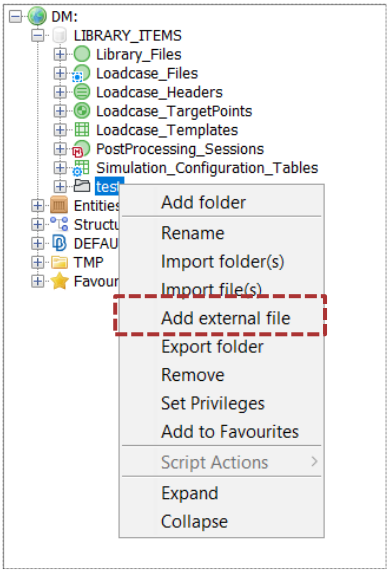


By double-clicking on an unverified External Source, a **Login** window appears where the credentials of the user must be given (**Username** and **Password**). If the credentials are valid, a dialog window gives the option to the user to store those credentials for as long as the session is open. By using this option, the user can add multiple external files without giving credentials for each one of them.

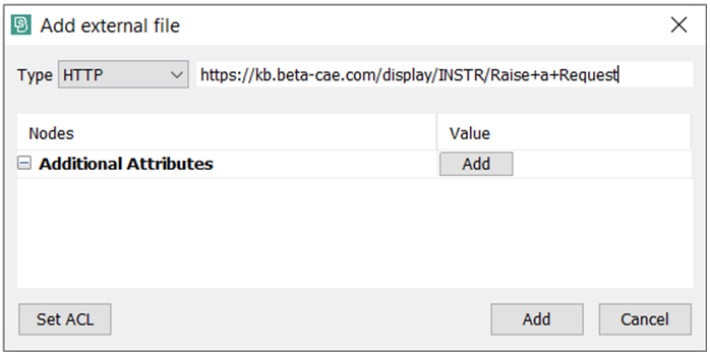


6.3.8.1. Add external file

After the External Source configuration, any file in terms of URL or mounted network share can be added in the Data Tree. The addition can take place from the context menu of any folder in the Data Tree.

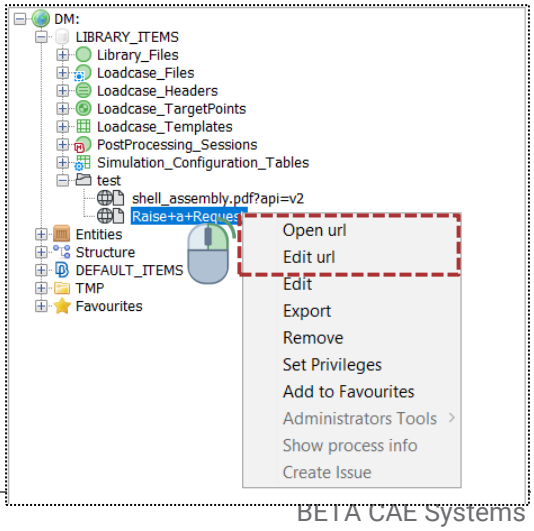


By selecting **Add external file** from the context menu, the **Add external file** window appears. The URL or the network path respectively can be defined. **Additional Attributes** can be added for the external file by the user.



If everything is valid, the external file will be added to the Data Tree and two options related to the functionality will appear in the context menu:

- **Open url** – Open the URL in web browser or file manager.
- **Edit url** – Edit the URL or the path to the network file

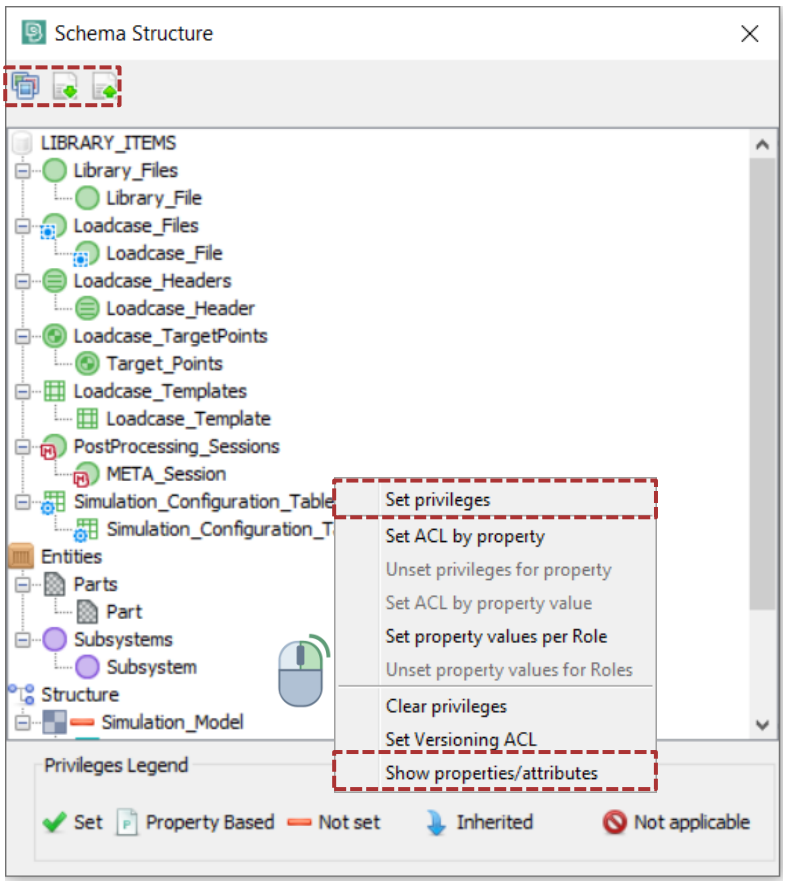


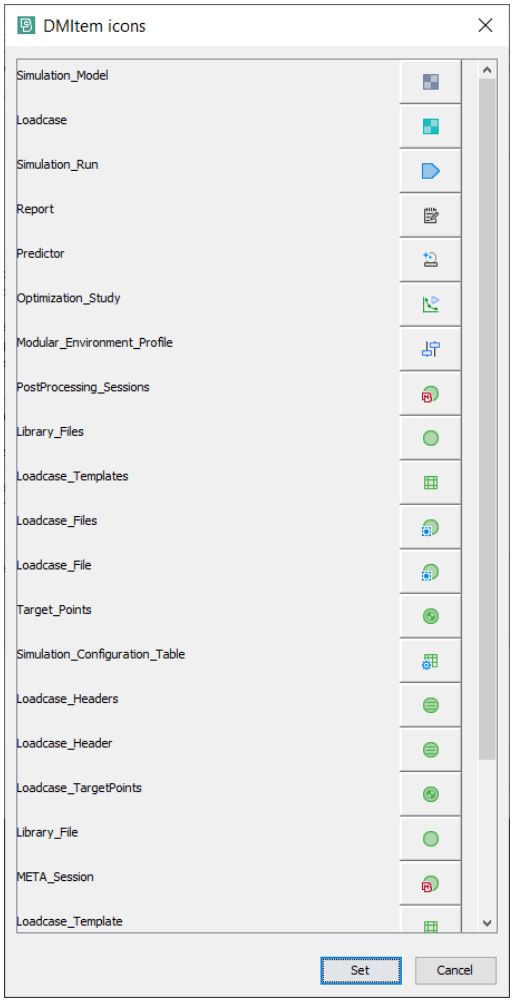
7. Administration tools

7.1. DM Schema Structure

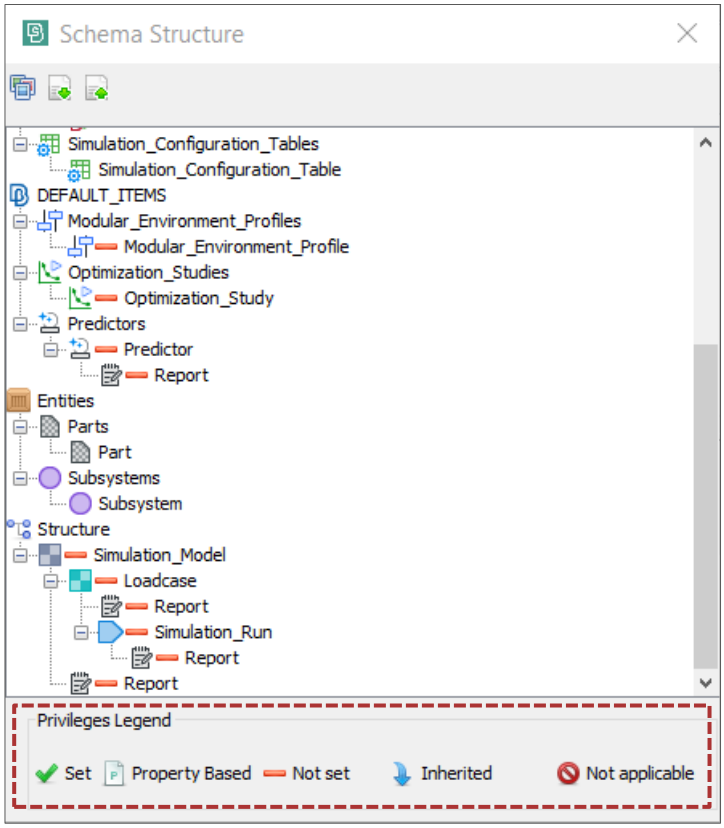
The Data Model can be previewed through *Data > DM Structure (Schema)*. In the **Schema Structure** window all the data types comprising the Data Model, along with their hierarchical relationships are schematically shown. The permissions of each data type under the Structure container can be defined.

The icon of each data type can be specified through the **DMItem icons** window accessed by pressing the **SetDMIcons** button 





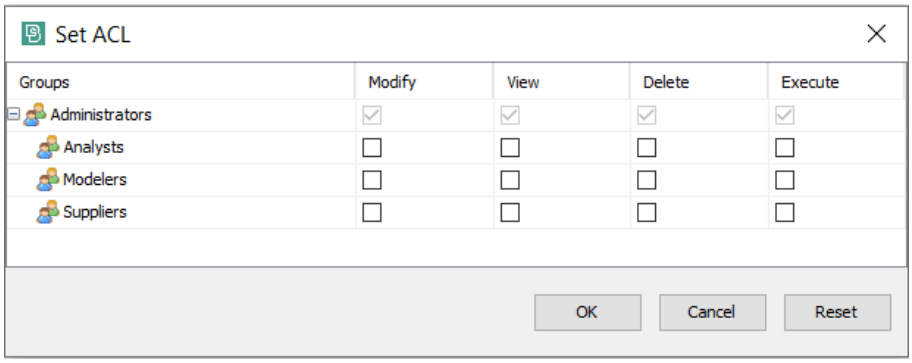
The icons and ACLs of all data types can be exported/imported through the respective buttons  .



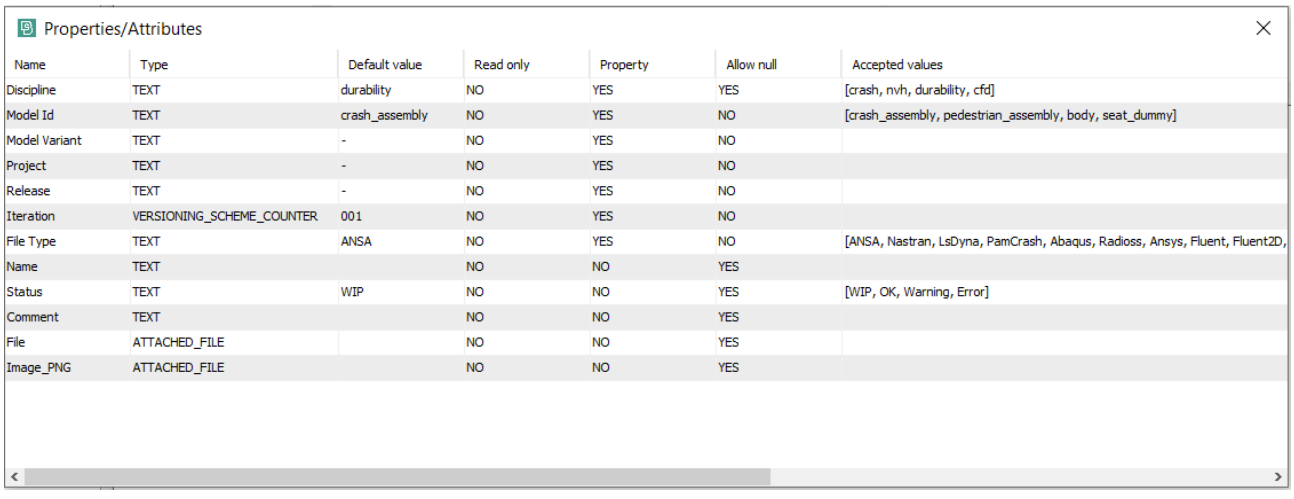
For each data type, the status of privileges is visible:

- Privileges have been specified
- Specified privileges are property based
- Privileges are inherited from the parent data type
- Privileges are not applicable

Selecting the **Set privileges** option of the context menu, the **Set ACL** window appears where the administrator can set privileges for the selected data type.

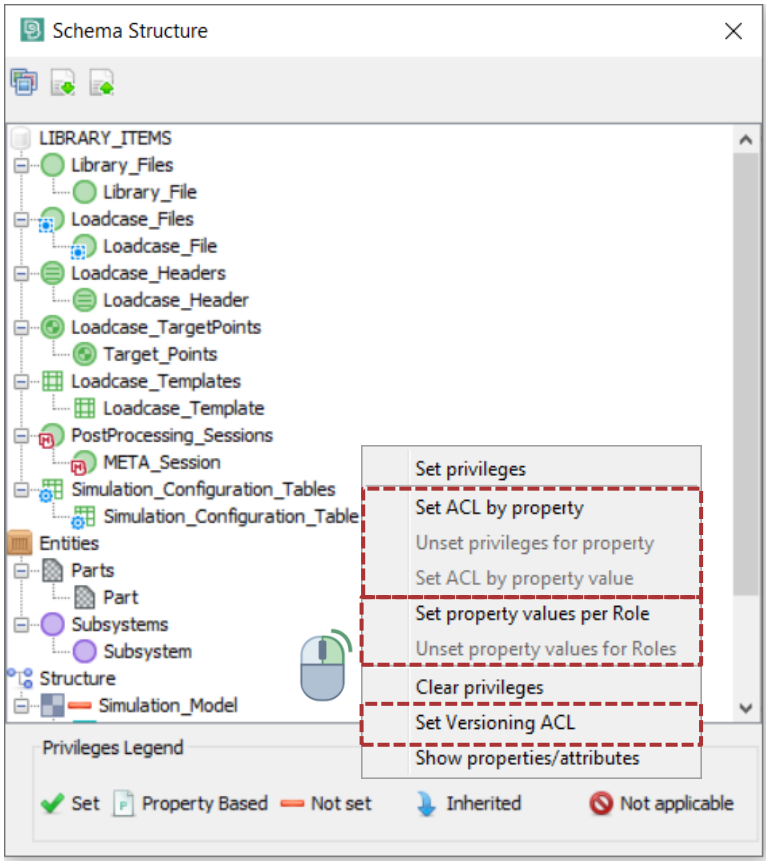


Selecting the **Show properties/attributes** option of the context menu of a selected data type, the **Properties/Attributes** window appears.

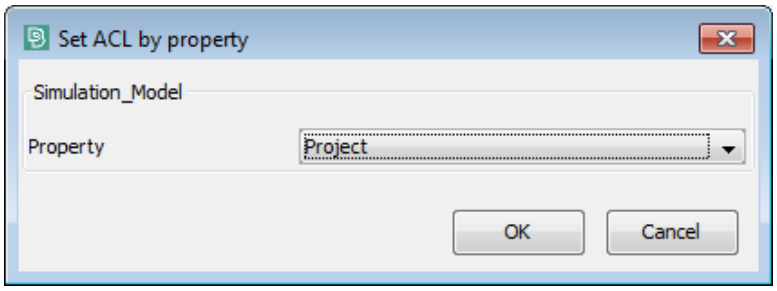


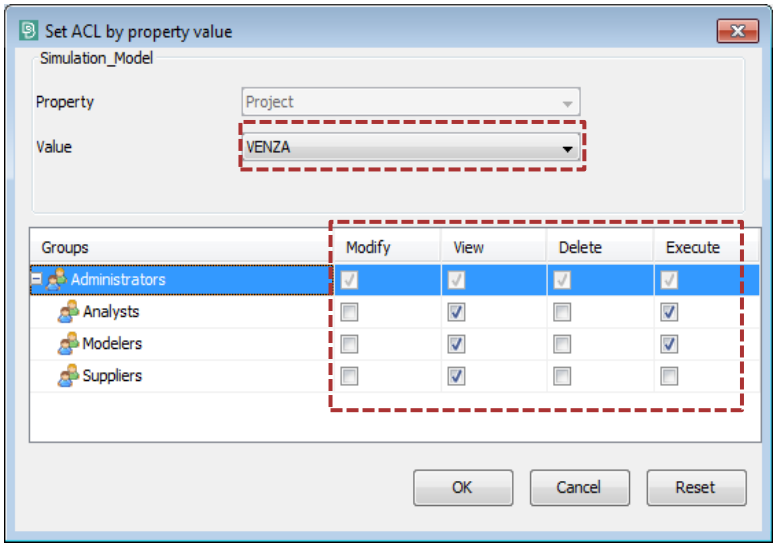
Apart from specifying general ACLs per data type, more advanced options are available through the DM structure (Schema) function:

- Specify privileges per property value
- Control which property values are available to each user role
- Specify the versioning scheme permissions

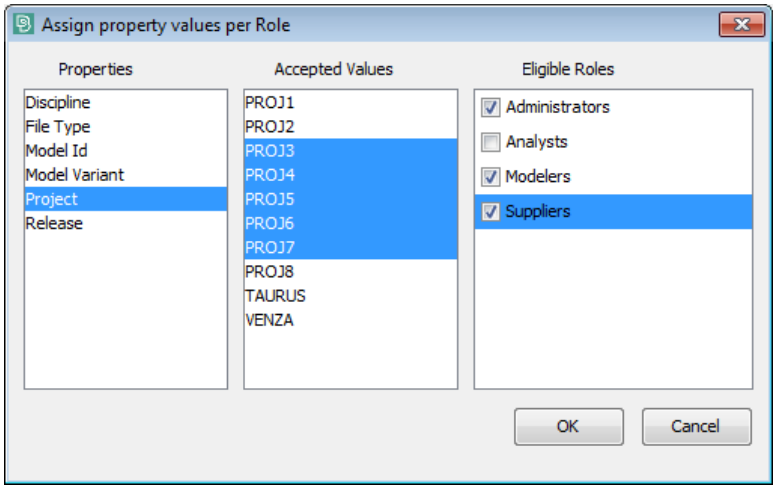


Set privileges per property option: The specified privileges are applied to all the existing data objects of the selected data type and property value.

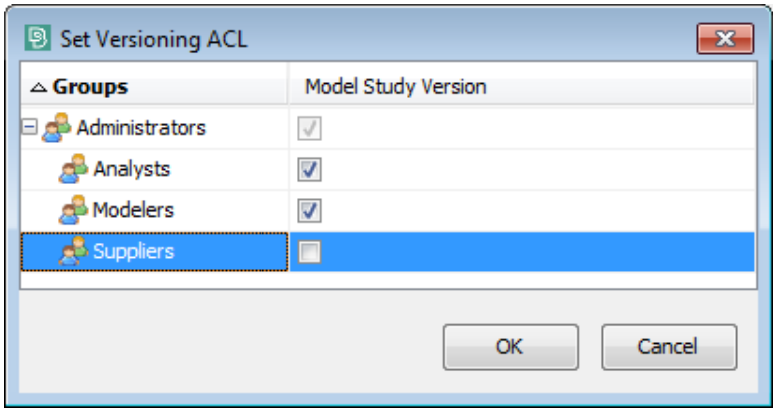




Set property values per Role option: During the creation of a new object, the selected accepted values will be available only for the specified roles.

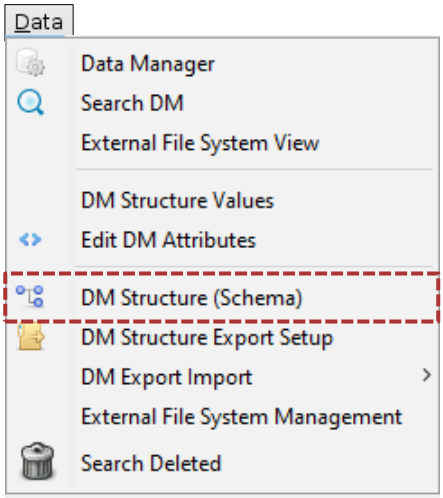


Set Versioning ACL option: Specify the user roles which are allowed to spin up each versioning property of the selected data type.

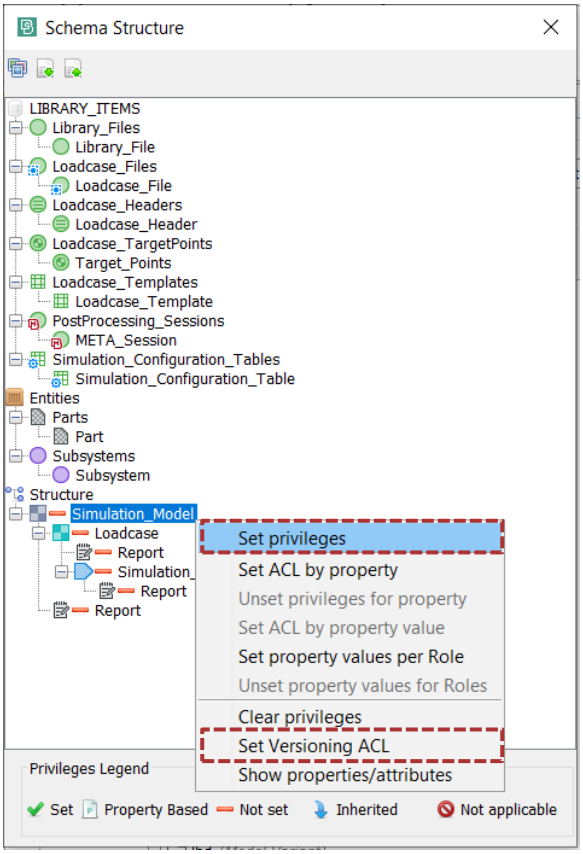


7.1.1. Data Security

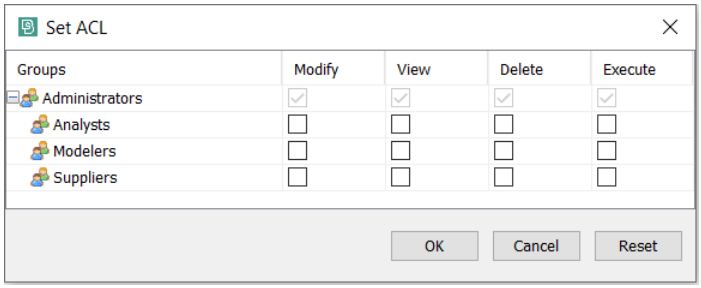
In SPDRM, privileges on data are set with the aid of Access Control Lists (ACLs) which essentially control which user roles will be able to View, Modify, Delete or Execute particular data objects. For example, if the user's role doesn't have view privileges on a subsystem, it won't be possible to find it in the Search workspace. Similarly, without Modify privilege, it won't be possible to edit its file or modify its attributes and without Delete privileges it won't be possible to delete it.



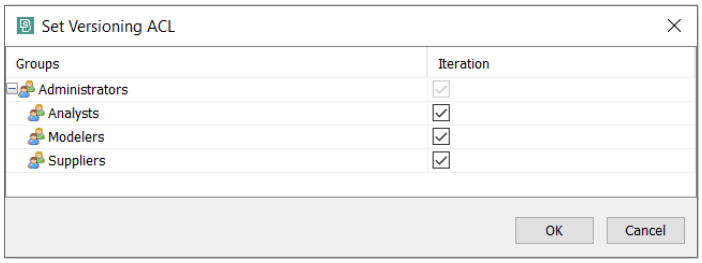
Default access control rules are defined in the server configuration file (*taxis.conf*). When it comes to data security in particular, generic access control rules can be set on DM item types. These rules control the default privileges that will be assigned to newly created DM items according to their type. To set the permissions on DM item types, the user can invoke the *Schema Structure* window through **Data > DM Structure (Schema)** from the SPDRM client menu bar.



In the *Schema Structure* window, the user can right click on the desired type and select the option **Set privileges**. This will invoke the *Set ACL* window where permissions for each group/role can be defined using the respective checkboxes.

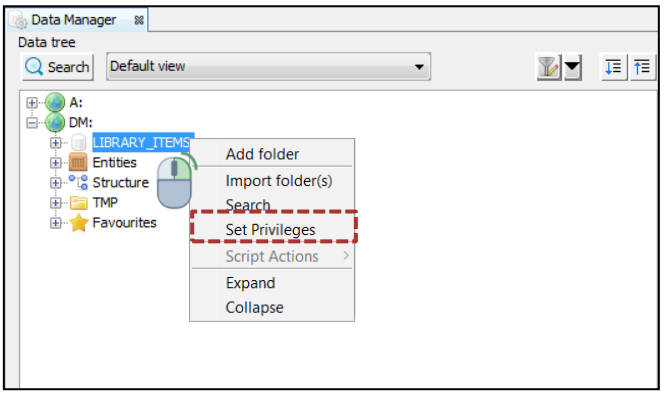


Similarly, using the option **Set Versioning ACL**, it is possible to define which user roles can spin-up particular versions of the selected DM object type.

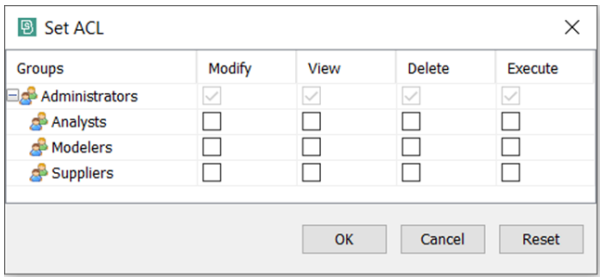


Access control can also be set on DM root items (i.e. *LIBRARY_ITEMS*, *Entities*, *Structure*) from the *Data Manager* window, to control who will be able to add, modify, view, or delete files, folders and/or DM items.

For example, if the user's role doesn't have view privileges on a folder, it won't be possible to view it in the DM Data tree. Similarly, without Modify privilege, it won't be possible its attributes. Without Delete privileges it won't be possible to delete it, while without Execute privileges it won't be possible to access its contents.



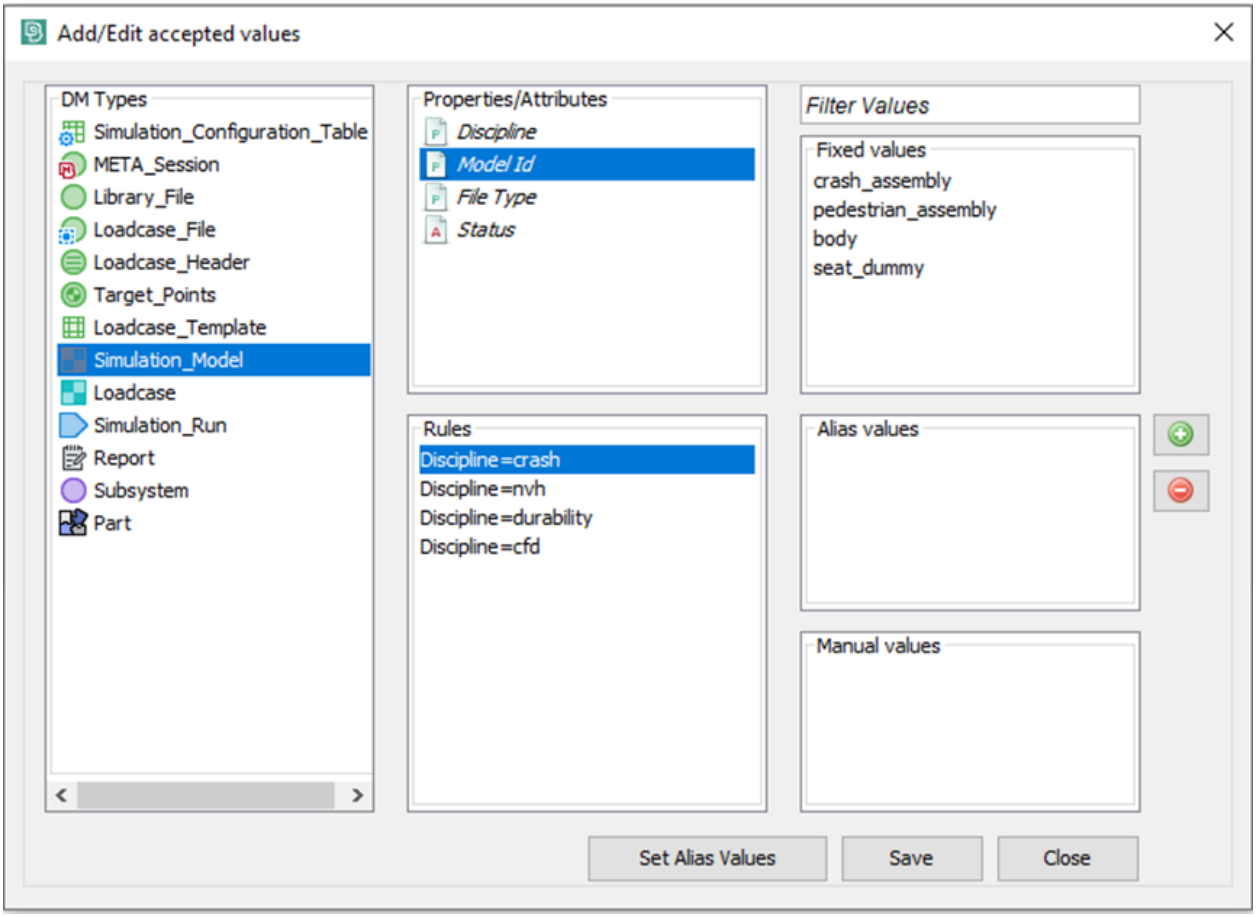
To define the privileges on DM root items, the user can select the option **Set Privileges** from the context menu in the Data Manager Tree and define the permissions for each group/role using the respective checkboxes in the *Set ACL* window that appears.



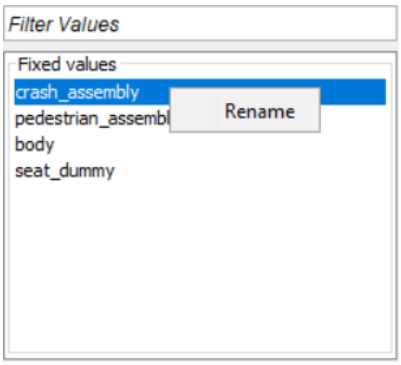
7.2. DM Structure Values Configuration

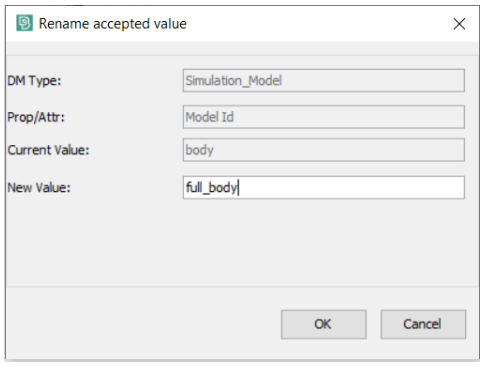
The Data Model of an SPDRM environment is defined in the **DM_structure_TBM.xml** file. Each data type is defined by a number of Properties/Attributes that are often determined by Rules and Accepted Values.

The **Add/Edit accepted values** tool, through **Data > DM Structure Values**, enables configuring those Rules/Accepted Values by the SPDRM Client, without modifying the DM_structure_TBM.xml file manually.

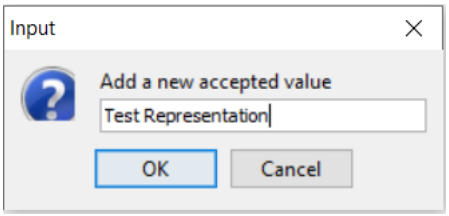


The main window lists all DM Item Types defined in Data Model alongside the Properties/Attributes, Rules and Accepted Values for each type.





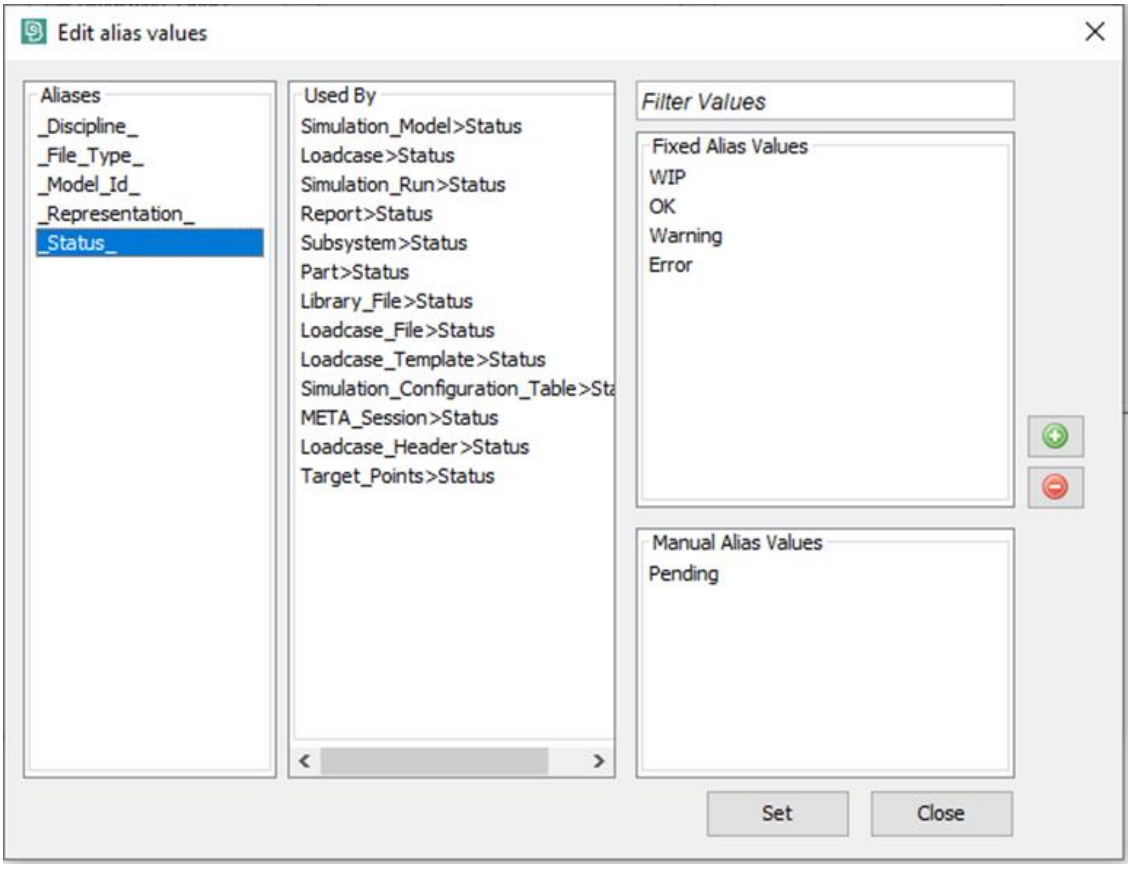
A defined Fixed or Alias value can be renamed by the context menu of the Fixed or Alias Values list.

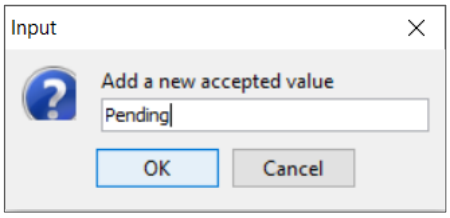


A new manual value can be added for the Property/Attribute by pressing the Add button. By using the Remove button the previously added Manual value can be removed.

After pressing the OK button a direct update of the value is performed.

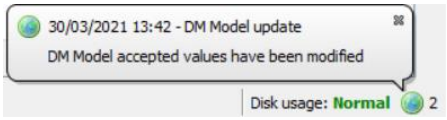
There are some Alias Values in the Data Model that are defined a single time and are used by multiple DM Items. By selecting **Set Alias Values** in the tool's main window, a new window named Edit Alias Values appears where all the Alias Values are listed.





A new manual value can be added for each of the Property/Attribute by pressing the Add button. By using the Remove button the previously added Manual value can be removed.

After pressing the Set button the update of the Alias value is performed.



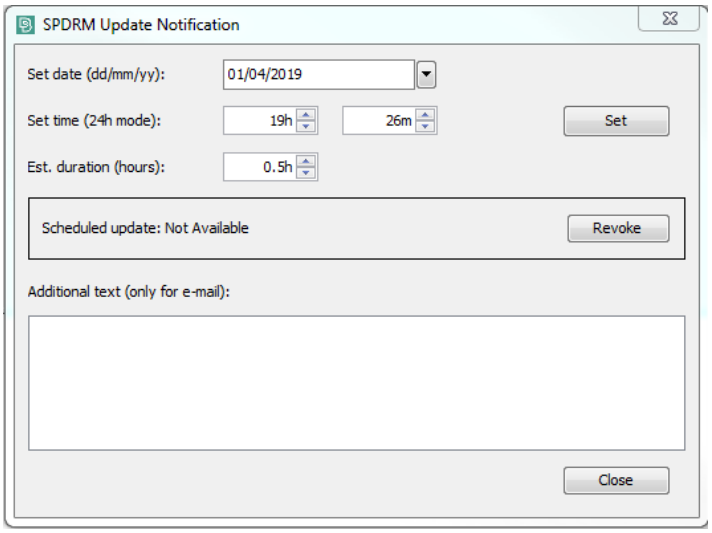
The confirmation of all the performed changes happens when the Save button in the main windows is pressed. All changes are available for the creation of new items, just after the confirmation balloon appears at the right bottom corner of the SPDRM Client.

7.3. SPDRM Update Notification

SPDRM Update Notification mechanism provides the option to SPDRM Administrators to inform the SPDRM Users about a forthcoming SPDRM update/maintenance task.

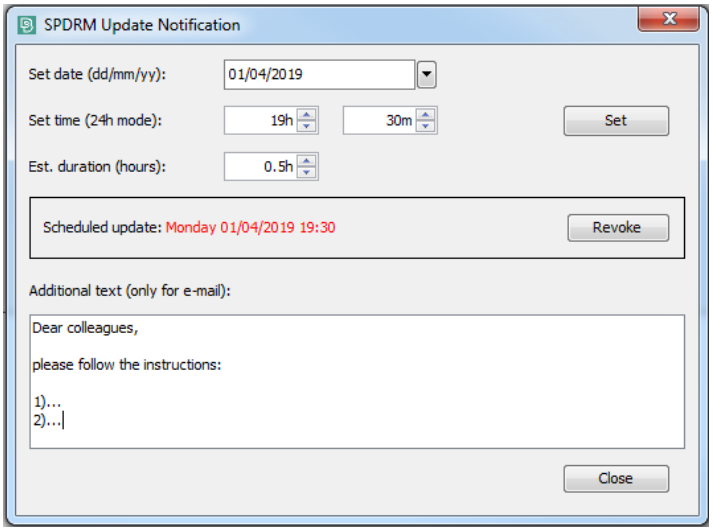
This feature informs the SPDRM users about the forthcoming SPDRM update in order to plan their work appropriately and close their client sessions before the scheduled update time.

Go to **Tools > SPDRM Update Notification** to access the functionality.

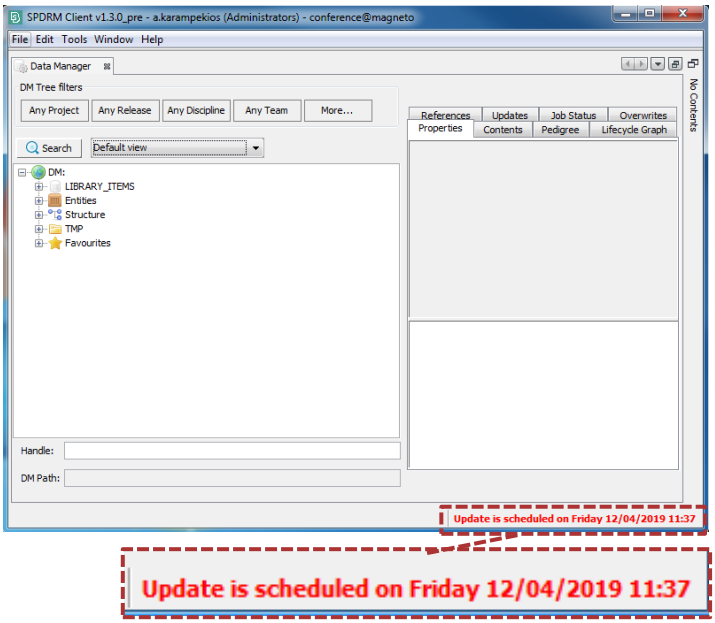


In this tool the exact date, time and duration of the update can be set. Moreover, a custom message can be optionally set in the **Additional text (only for e-mail)** field , which will be sent to the users via e-mail.

By pressing the **Set** button, the **Scheduled update** field is updated with the exact update info.



After an SPDRM Update is scheduled, a notification/flash message is displayed at bottom right corner of the SPDRM client window. The message remains active up to the specific date and time displayed.



Apart from the notification/flash message, an e-mail, containing a default message, plus the additional text that has been set in previous step (if any), is sent auto matically to all SPDRM users, informing them about the upcoming update.

```
SPDRM Update is scheduled by user a.karampekios on Monday 01/04/2019 19:30 for 0.5 hour(s)

Please note that this time is the server local time.
Login to your SPDRM Client to get the notification in your local time.

Additional comments from user:
Dear colleagues,

please follow the instructions:

1)...
2)...
```

The SPDRM Administrators have also the option to cancel a scheduled update by pressing the **Revoke** button, at the configuration window. On **Revoke**, a new notification/flash message appears.

```
Scheduled SPDRM update has been cancelled by user a.karampekios
```

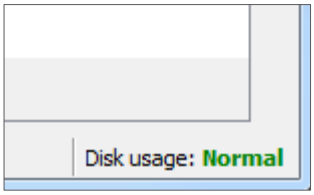
At the same time, an e-mail is sent to all the users automatically, informing them about the update cancelation.

7.4. Audit Mechanism

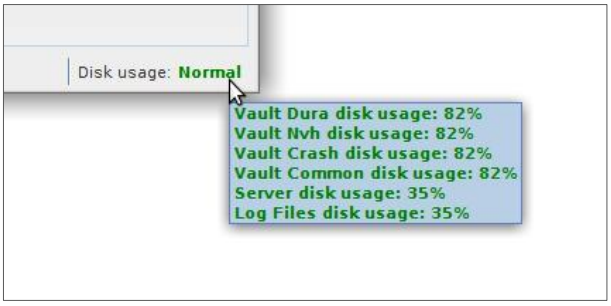
Audit Mechanism is a mechanism available in SPDRM that enables monitoring of all vaults, server and log files file-systems directly through SPDRM Client.

A message regarding the file-systems' status is constantly shown at the status line (bottom right corner) of every SPDRM client. A tooltip is displayed when the user points with mouse on the status line message.

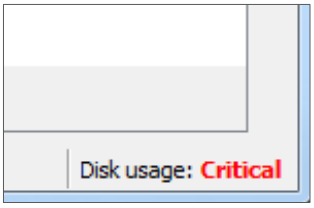
Audit mechanism, when enabled, performs a periodic check of the file-systems and updates their status every **3 hours**.



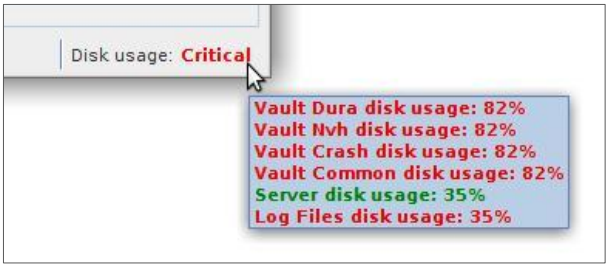
Status line message when file-systems disk usage is Normal.



Tooltip message when all monitored file-systems are below threshold.



Status line message when file-systems disk usage is Critical.



Tooltip message with mixed Critical and Normal file-systems disk usage.

7.4.1. Configuration

The above mechanism is enabled when a file named **audit.conf** is added to the SPDRM configuration files directory. This is the directory where **taxis.conf** and rest configuration files are located. In case the referred file is absent the Audit Mechanism is deactivated. The syntax of that file should have the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
<comment/>
<entry key="vaultDiskUsageThreshold">90</entry>
<entry key="serverDiskUsageThreshold">90</entry>
<entry key="logFilesDiskUsageThreshold">90</entry>
</properties>
```

As it can be seen in the format, 3 keys are currently supported:

1. vaultDiskThreshold
    - Enables the monitoring of the vaults disk usage.
    - All vaults that have been configured in *taxis.conf* file are checked.

- Each vault's status is reported separately.
- Threshold is common, however, for all vaults.

2. serverDiskUsageThreshold
    - Enables the monitoring of the server disk usage.
    - What we actually check is the file-system of the running domain.
3. logFilesDiskUsageThreshold
    - Enables the monitoring of log files disk usage.

NOTES:

- The value of each key represents the percentage of usage (threshold) above which the status of the disk usage should be conceived as 'Critical'.
- Any value of the actual disk usage below the threshold is conceived as 'Normal'.
- Accepted value for all keys is an integer between **20** and **95**. Any other value disables usage monitoring of respective disk.
- Audit mechanism works also in case of multi-site environments if a RIOC server is running in every remote site. When an administrator logs-in to a remote client, information about disk usage in all sites will be displayed, while in case of "non-admin" users, only the disk usage of the current site is displayed.

7.5. Automatic clean-up of remote vaults (Multi-site)

Audit mechanism configuration file (**audit.conf**) can also be used for enabling the automatic clean-up mechanism of the remote vaults. This is a mechanism for data purging on remote vaults that serve as file caching servers on remote vaults. This mechanism is responsible for the removal of not recently used files located in remote vaults (**firstDir**), in order to save disk space on the remote file-systems and to create free space for new files that need to be replicated from the main site. This mechanism provides configuration for the following:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
<comment/>
<entry key="cleanUpRemoteSites">India,China,Pakistan</entry>
<entry key="India_cleanUpMaxStorageThreshold_Common">3G</entry>
<entry key="India_cleanUpMaxStorageThreshold_Crash">1G</entry>
<entry key="China_cleanUpMaxStorageThreshold_Common">100M</entry>
<entry key="Pakistan_cleanUpMaxStorageThreshold_Common">2G</entry>
<entry key="India_cleanUpRule_Common">MAX:450M,MIN:10M,CTIME:50</entry>
<entry key="India_cleanUpRule_Crash">MAX:2G,MIN:200M,ATIME:100</entry>
<entry key="China_cleanUpRule_Common">MAX:50,MIN:1M,CTIME:20</entry>
<entry key="Pakistan_cleanUpRule_Common">MAX:550M,MIN:50M,CTIME:40</entry>
</properties>
```

1. **Name**
   - Defines the name of the remote sites for which the mechanism should be enabled (given that the file caching mechanism is enabled)
   - Respective key is cleanUpRemoteSites
   - e.g.<entry key=cleanUpRemoteSites”>India,China,Pakistan</entry>
2. **Threshold**
   - Defines the max storage on the disk, defined per site and per vault
   - Respective key is <site>_cleanUpMaxStorageThreshold_<vault_name>
   - Value is an integer followed by M for Megabytes and G for Gigabytes
   - e.g.<entry key=“India_cleanUpMaxStorageThreshold_Crash”>1G</entry>
3. **Purging rule**
   - Defines the rules that will be applied for the deletion of the files
   - Respective key is <site>_cleanUpRule_<vault_name>
   - Valid values are ***MAX*** for maximum size of searched files, ***MIN*** for minimum size, ***CTIME*** for the number of days that the searched files haven’t been changed and ***ATIME*** for the number of the days that the files haven’t been accessed. Rules must be defined using comma separation.
   - e.g <entry key=“India_cleanUpRule_Crash”>MAX:2G,MIN:200M,ATIME:100</entry>
   - The rule above is interpreted to the following command

     ```
     find /path/to/vault/firstDir/ -type f -size +200M -size -2G -atime +100
     ```

NOTES:

- If the purging rule fails to remove sufficient amount of data so as to drop the disk usage under the defined threshold, then a fallback mechanism will be involved to drop the disk usage below 80% of the defined threshold. This mechanism will delete files in the vault (firstDir) in descending access time order.
- The mechanism is triggered every Sunday at 02:00 AM server’s time and check vault’s disk usage in order to decide whether it will be involved or not.

8. Maintenance operations

8.1. SPDRM Backup

It is important to frequently back up (e.g. daily) the database, configuration files and vault to prevent data loss in case problems occur, such as system crashes, hardware failures, or users deleting data by mistake.

Backups are also essential as data security measures before performing an update on the database, or the SPDRM Server/Client installation.

Additionally, they can be used to transfer data from an existing SPDRM installation to another, or to set up replication slave servers.

It is suggested that during the backup and recovery of the database, the database should be up and running, but all SPDRM client instances should be closed, and the SPDRM Server should be stopped.

If this is not possible (e.g. due to overnight batch jobs, or multi-site installation with different time zones in working hours of the users), then the backup operation should take place when the minimum number of transactions are running on the server.

NOTES:

- The system will be never in an incoherent state even in the case that the database will be backed-up, while there are running transactions.
- In case that the backup of the database and the vault (firstDir) will take place in an asynchronous manner, then the backup of the database should be performed first, and the backup of the vault (firstDir) should follow.
- The database is not corrupted because files in the vault (firstDir) are always added, and never removed by the system.
- The system returns at the state where it was when the database backup has been done.
- The files in the vault (firstDir) created after the backup and before the recovery date will become “orphan”.
- The “orphan” files in the vault (firstDir) can be optionally removed manually.
- Only work done within the date of backup and the date of recovery will be lost.

8.1.1. Database

8.1.1.1. MySQL

1. Login to the machine where MySQL database is installed.
2. Open a terminal window and change directory to: [MYSQL_INSTALL_DIR]/bin (e.g. cd /opt/mysql/bin/).
3. Run the command:
   ```
   ./mysqldump -u root -p taxisdb > /path/to/YYYYMMDD_taxisdb.sql
   ```
4. Enter the password for the MySQL root user.

8.1.1.2. Oracle

1. Login to the machine where Oracle database is installed.
2. Open a terminal window and change directory to the [ORACLE_INSTALL_DIR]/BIN (e.g. cd /opt/oracle/product/11.2.1/dbhome_1/BIN/).
3. Run the command:
```
./expdp SYSTEM schemas=SPDRM directory=DATA_PUMP_DIR
dumpfile=YYYYMMDD_expdp_spdrm.dmp logfile=YYYYMMDD_expdp_spdrm.log exclude=user
```
4. Enter the Oracle Administrative (SYS) password.

8.1.2. Configuration & Vault

SPDRM Server Configuration Backup

Some configuration files of the SPDRM Server keep relationships among the data model and the database tables.

Thus, in order to keep a full backup of the SPDRM data objects it is necessary to keep a backup of the files that are included in: [SPDRM_SERVER_DIR]/wildfly/standalone/configuration/.

SPDRM Client Configuration Backup

The following configuration and customization files of the SPDRM Client need to be .

- `[SPDRM_CLIENT_DIR]/dmicons/`
- `[SPDRM_CLIENT_DIR]/taxisprops.xml.linux`
- `[SPDRM_CLIENT_DIR]/taxisprops.xml.windows`

SPDRM Vault Backup

The *firstDir* subdirectory, which is located into the *vault* directory, is accessible only by the SPDRM Server and it is used for internal files storage and retrieval. All the physical files that are associated with the data objects that are saved in SPDRM are stored in the *firstDir* subdirectory.

All SPDRM data objects that are described by a physical file (e.g. Parts, Subsystems, Simulation Models, Runs, etc.) keeps a link between the database entry and the respective file in *firstDir* subdirectory.

Thus, in order to keep a full backup of the SPDRM data objects it is necessary to keep a backup of the files that are included in the *firstDir* subdirectory, along with the backup of the database and the server's configuration files.

8.2. SPDRM Recovery

8.2.1. Database

8.2.1.1. MySQL

1. Stop the SPDRM server.
2. Login to the machine where MySQL database is installed.
3. Open a terminal window and change directory to the [MYSQL_INSTALL_DIR]/bin (e.g. cd /opt/mysql/bin/).
4. Run the command:

```
./mysql -u root -p taxisdb < /path/to/YYYYMMDD_taxisdb.sql
```

5. Enter the password for the MySQL root user.
6. Start the SPDRM server.

8.2.1.2. Oracle

1. Login to the machine where Oracle database is installed.
2. Open a terminal window and change directory to the [ORACLE_INSTALL_DIR]/BIN (e.g. cd /opt/oracle/product/11.2.1/dbhome_1/BIN/).
3. Run the command:

```
./impdp SYSTEM remap_schema=SPDRM:SPDRM directory=DATA_PUMP_DIR
remap_tablespace=SPDRM_TABLESPACE:SPDRM_TABLESPACE
dumpfile=YYYYMMDD_expdp_spdrm.dmp logfile=YYYYMMDD_impdp_spdrm.log
```

4. Enter the Oracle Administrative (SYS) password.

8.2.2. Configuration & Vault

SPDRM Server Configuration Recovery

In case that the files of the server's *config* directory have been deleted or corrupted, the files that are kept during the backup should be restored, while the SPDRM server is stopped, in order for the system to become fully functional again.

SPDRM Client Configuration Recovery

In case that the files of the client's directory have been deleted or corrupted, the files that are kept during the backup should be restored. The SPDRM client sessions need to be restarted, in order for the changes to take effect.

SPDRM Vault Recovery

In case that the files of the *firstDir* subdirectory are not deleted or corrupted, then only the "orphan" files can be optionally removed manually. Otherwise, the system will overwrite them anyway once new files with the same IDs will be associated with SPDRM data objects.

In case that the files of the *firstDir* subdirectory have been deleted or corrupted, the files that are kept during the backup should be restored, in order for the system to become fully functional again.

8.3. SPDRM bundle

The SPDRM Bundle is used to update an SPDRM environment to a pre-release version.

The SPDRM Bundle usually consists of the following files:

- sql_updates.sql
- Taxis.ear
- taxisprops.zip

8.3.1. Apply DB updates

The installation of MySQL updates is described in the following steps:

1. Stop the SPDRM server
2. Keep a backup of the database (i.e. `mysqldump -u root taxisdb > /PATH_TO/mysqldump.sql`)
3. Apply the MySQL updates that are included in the **mysql_updates.sql** file
   - Ensure that MySQL is up and running
   - Open a terminal window and change directory to: <MYSQL_INSTALL_DIR>/bin
   - Run the following command to apply the MySQL updates:

     ```
     ./mysql -u root -p taxisdb -v < /PATH_TO/mysql_updates.sql
     ```
4. Start the SPDRM server

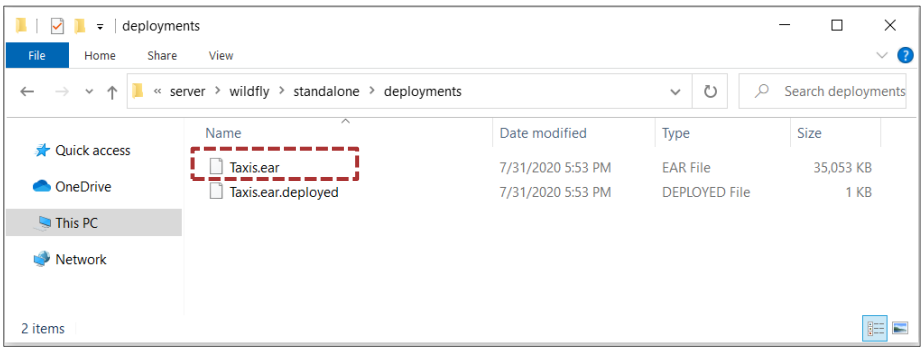
IMPORTANT: In order to install the required updates in MySQL, the MySQL database should be up and running, but all the SPDRM client instances must be closed, and the SPDRM Server should be stopped.

8.3.2. Update the SPDRM Server package (*Taxis.ear*)

8.3.2.1. Through the File Browser

The SPDRM Server update procedure is described in the following steps:

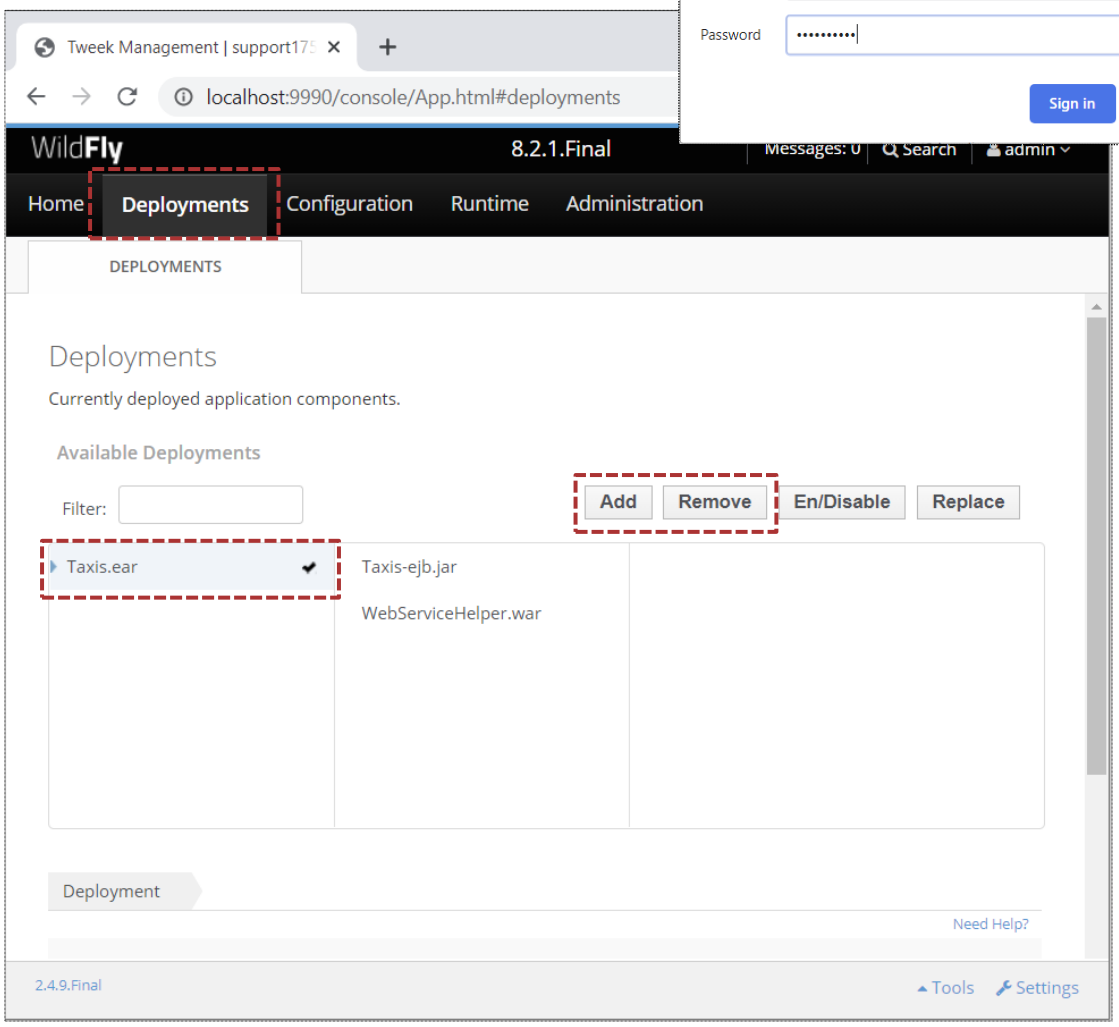
1. Copy the provided **taxis.ear** in:
   `[SPDRM_SERVER_DIR]/wildfly/standalone/deployments/`
   and replace the existing one.
2. The application server will automatically recognize the new server package (*taxis.ear*) and will soon start its deployment.
3. After finishing successfully the automatic deployment of the new server package, the contents of the deployments directory should look like the image at the right.



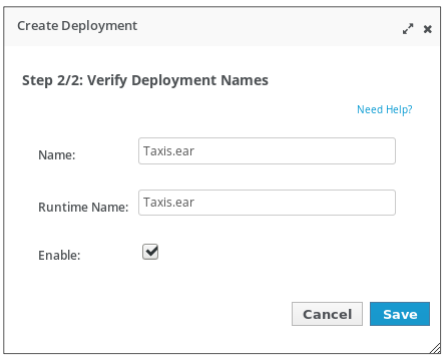
IMPORTANT: In order to update the SPDRM Server using a provided SPDRM server update package (*taxis.ear*), both MySQL and SPDRM Server should be up and running, but all the SPDRM client instances must be closed.

8.3.2.2. Through the WildFly admin console

1. Open a Web Browser and go to `http://<SPDRM_SERVER_HOST>:9990` (WildFly Admin console).
2. Login using the following credentials:
   - User Name: *admin*
   - Password: *adminadmin*



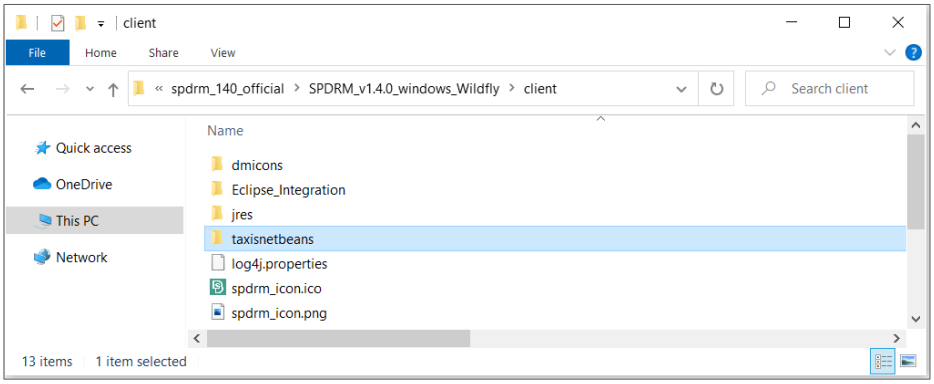
3. Switch to the **Deployments** tab
4. Select the *Taxis.ear* item from the list of current deployed application components
5. Press the **Remove** button to remove it
6. Press the **Add** button and select the new Taxis.ear
7. Check the **Enable** checkbox and press the **Save** button to deploy the new *Taxis.ear*



8.3.3. Update the SPDRM Client package (*taxisnetbeans.zip*)

The SPDRM Client update procedure is described in the following steps:

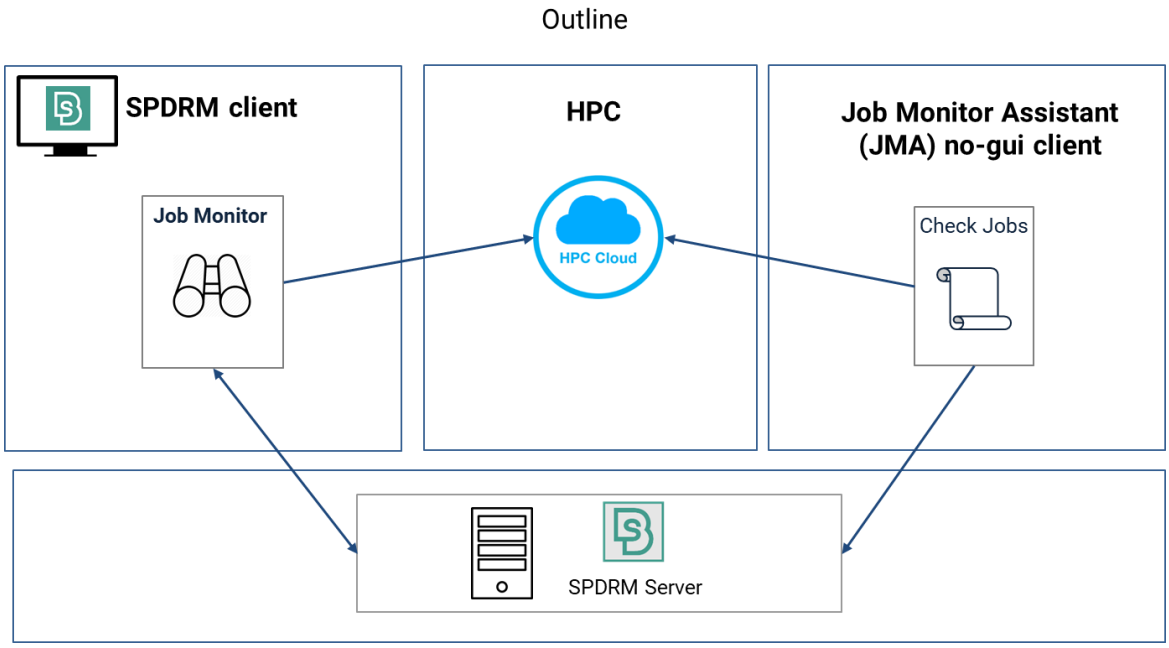
1. Delete the existing **taxisnetbeans** folder that is located under the client's directory (i.e. `[SPDRM_CLIENT_DIR]/taxisnetbeans/`
2. Extract all the contents of the new **taxisnetbeans.zip** file into the client's directory (i.e. `[SPDRM_CLIENT_DIR]/`



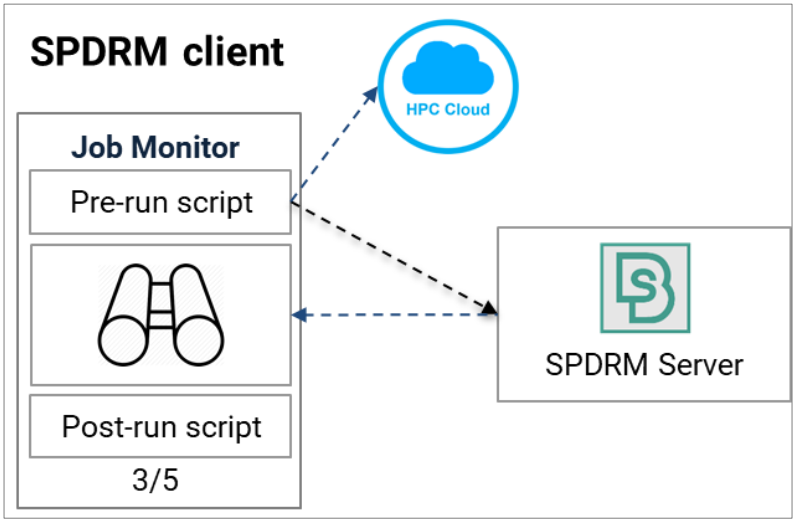
IMPORTANT: In order to update the SPDRM Client using a provided SPDRM client update package (*taxisnetbeans.zip*), all the SPDRM Client instances that are running on Windows OS must be closed.

9. HPC Jobs Management

9.1. Introduction



9.2. Submission & Monitor



Pre-run script

- Submits Job to HPC and get correlation id (or from input slot)
- Registers Job to server

Monitoring

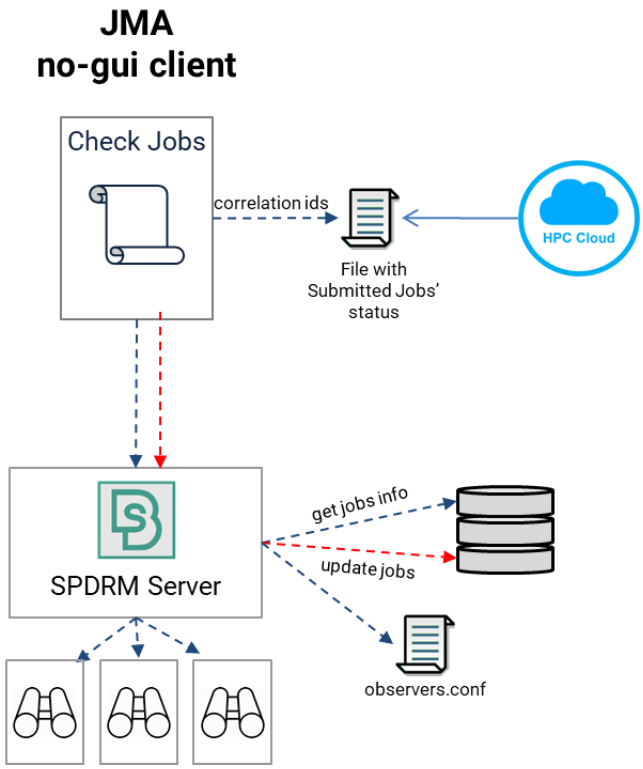
- Receives status events from server for correlated sub-tasks
- Supports cancel
- Shows jobs info table

Post-run script

- All sub-tasks info is available

Job Id	Correlation Id	Job Name	Status
100	corr1	DSI	DONE
101	corr1	BDW1	PEND

9.3. HPC Query & Update



Periodically

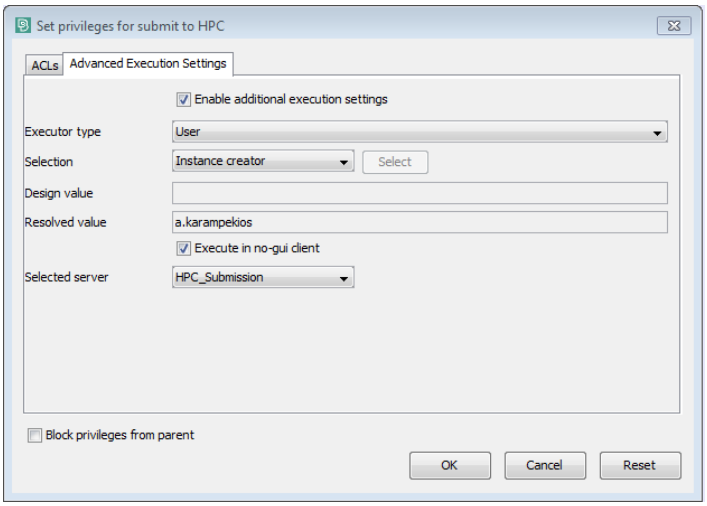
- Asks SPDRM Server for running jobs
- Reads Submitted Jobs' status from HPC exported file
- Updates SPDRM Server jobs status

SPDRM Server

- Translates jobs data
- Keeps track of all Job Monitor nodes
- Sends notifications to Job Nodes

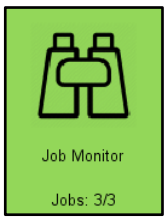
9.4. Submit in no-gui client

- If a no-gui client is already running for a particular user, then the same session will be used for the submission and monitoring.
- **Login to no-gui client** script node is schematically used in the previous slide for reasons of understanding. The start of the no-gui client and the execution of the node is set-up by the Privileges window of the Observer Node, by selecting the option **Execute in no-gui client**.
- In the same window, a registered Beta Apps Launcher, dedicated on executing specific jobs, can be defined with it's alias (e.g. HPC Submission).



9.5. Observer Node

Observer Node Definition



Finished Observer Node

Running Observer Node (No Job Completed)

Running Observer Node (One out of three Jobs Completed)

An observer node undertakes the monitoring of a job submitted to HPC.

There are two types of Observer Nodes, the **Script** and the **Application Observer Node**.

The **Script Observer Node** consists of three components: **Pre-run**, **Post-run** and **Cancellation** scripts

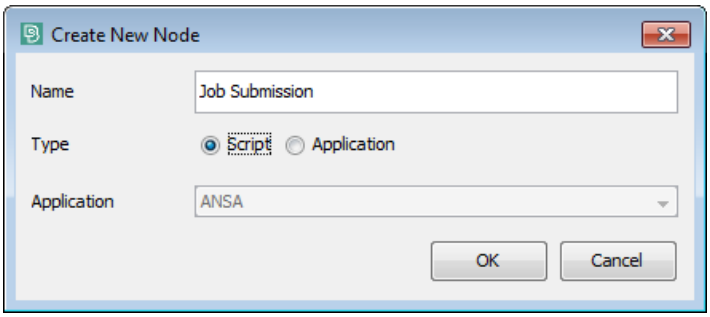
The **Application Observer Node** contains two extra components, the **Application** tab and the **Post-application** script. The Application tab can be used in the same way that an application node is used, in order to set up a 3rd party application execution. That is necessary when HPC submission takes place by using such application (e.g. Windows PowerShell)

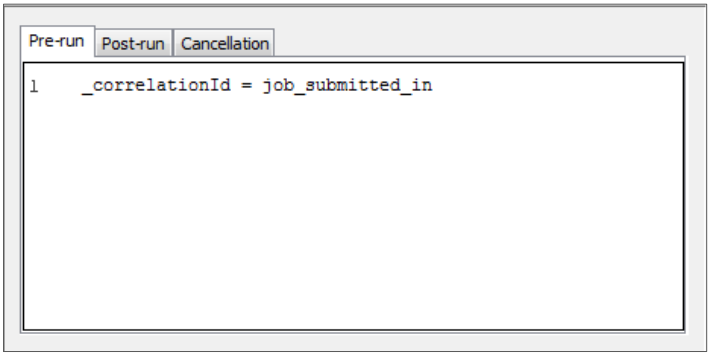
Obligatory input for the monitoring job is the identity of the submitted job, which is set to the build-in variable of the Observer Node, called **_correlationId**. The correlation id must be set in the pre-run script (for Script types) or the post-application (for Application types) of the Observer Node. It is the identity of each Observer Node running and through this the node takes updates about theirs jobs from the JMA (no-gui client) that communicates with the HPC.

9.6. Script Observer Node Editor

When Observer Node is selected from the **Palette** to be created, the **Create New Node** appears.

Image in the right shows the menu that appears where **Name** and **Type** of the Observer Node must be selected. If **Application** type is selected, the Application drop down menu is enabled, from which a registered application can be selected.





Script Observer's node editor contains 3 tabs:

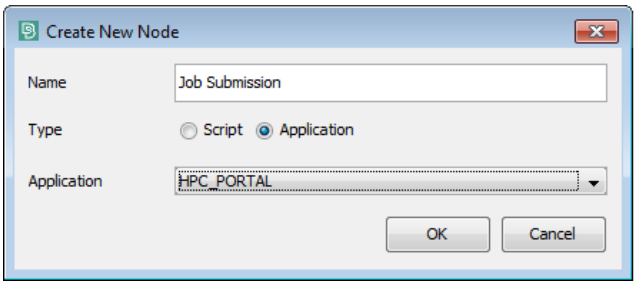
1. Pre-run script, where **_correlationId** variable must be obligatory set.
2. Post-run script, where all sub-tasks info is available.
3. Cancellation script.

Between pre- and post-run script, the observer node is in monitoring state.

A build-in variable named **_jobs** can be used in post-run script, that returns a list with all the jobs objects in the node. Those objects have the following attributes:

1. uniqueId
2. correlationId
3. status
4. jobAttributes

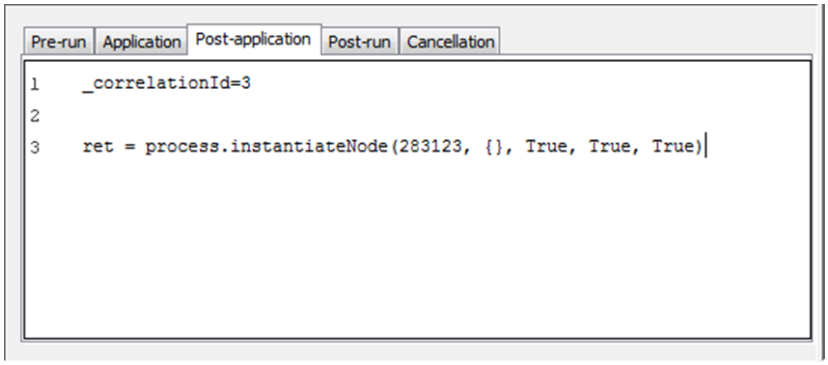
9.7. Application Observer Node Editor



The Application Observer node in the Visual panel contains an icon that indicates the registered application it executes.

The Application Observer's node script editor contains 5 tabs:

1. **Pre-run** script, where all preparation for the application takes place.
2. **Application**, where the set-up command is executed.
3. **Post-application**, where the _correlationId must be defined.
4. **Post-run** script, where all sub-tasks info is available.
5. **Cancellation** script.

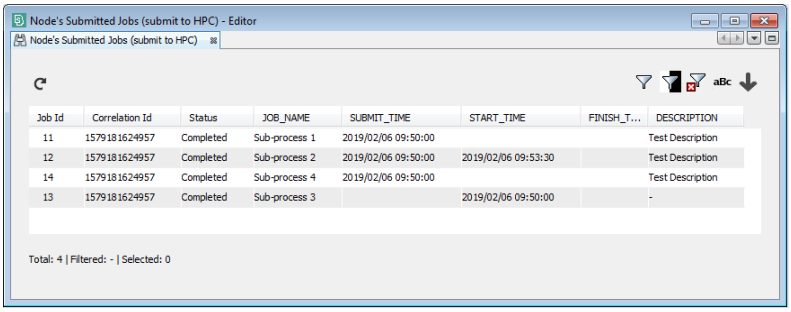


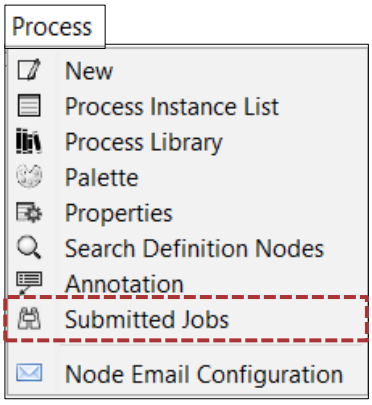
Between the post-application and the post-run script, the observer node is in monitoring state.

9.8. Jobs Monitoring

Submitted Jobs option has been added in the context menu of the Observer Node, which opens the **Node's Submitted jobs(<node name>)** window. Node's Submitted jobs window contains:

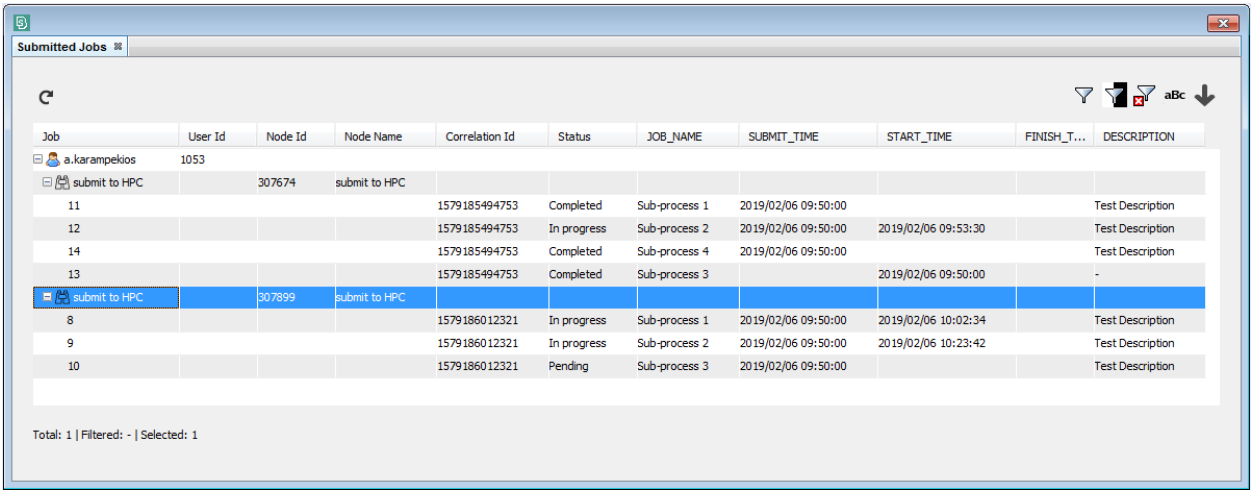
- Information coming from the JMA (no-gui client) (e.g. Status, Job Name, Description).
- Columns that are displayed are configurable and can be set in the **observers.conf** configuration file.
- Status updates of the ongoing jobs can be succeeded real-time by using the **Refresh** button.





The user can have an overview of all the submitted jobs per user through **Process -> Submitted Jobs**

In the **Submitted Jobs** window there is a list with all the users with running jobs and an attached list with all observer nodes in observing state for each one of them.



9.9. JMA no-gui Client

9.9.1. Configuration File

JMA's no-gui client job is to periodically ask the SPDRM Server for running jobs, read Submitted Jobs' status from HPC exported file and update the SPDRM Server jobs' status.

JMA no-gui Client is managed exclusively by the SPDRM Administrator who is responsible for:

- Starting and stopping the client.
- The script that will be executed periodically and will update the observer nodes.

The script function that sends information to all observer nodes, by reading the HPC's .json file is:

```
process.updateObservableJobs(full_path_to_hpc_json_file)
```

Configuration File:

- HPC Submission implementation requires a special configuration file, which is named **observers.conf**, that must be located in SPDRM's server configuration files directory.
- **observers.conf** is needed, so that a mapping is performed between HPC's status options and SPDRM build-in status options .
- In **observers.conf** the columns with information and their value types, which are contained in HPC's .json file, are defined in order to be displayed in the Jobs Monitor tool.
- Finally, the states that define the completion of a submitted job can be defined in **observers.conf**

```
#the name of the column that indicates correlation id (more than one jobs
#that were emerged from the same initial request have the same correlation id)
correlation.id.key = JOBID
#the name of the column that indicates unique id of the job
unique.id.key = SUBJOBID
#the name of the column that indicates status
job.status = STATUS

#the attributes of the job that will be stored in SPDRM
job.attrs = NAME, USER, START_TIME, CPUs

#the mapping of the attributes types
#accepted values are: string,int,boolean,float,date
job.attr.NAME = string
job.attr.USER = string
job.attr.START_TIME = date
job.attr.CPUs = string

#the pattern of the date values found in the csv/json (see java.util.Formatter.java)
date.pattern = yyyy\/MM\/dd HH:mm:ss


#the mapping of the various custom to in-house statuses

status.PENDING = PENDING
status.IN_PROGRESS =  RUNNING
status.CANCELLED = CANCELLED
status.ERROR = ERROR
status.COMPLETED = FINISHED

completed.states = FINISHED, CANCELLED
```

observers.conf Sample File

9.10. JMA Package

The required files are listed in the table below.

File	Description
start_jma.sh	A shell script that runs a no-gui client and executes the JMA process.
stop_jma.sh	A shell script that stops the no-gui client that is opened by start_jma.sh.
jma_instantiator.py	A python script that is used to instantiated the JMA.
jma.conf	A configuration file where the path to the file with Submitted Jobs' status from HPC is set.
JMA_Process.json	The JMA process that updates the Observer Nodes.
HPC_Job_Monitor_sample.json	A sample version of the HPC Job monitor process.

NOTE: All files are contained in the same folder (**jma**) copied by the SPDRM installer in the parent folder of the SPDRM Client.

NOTE: An extra file named *running_jobs_sample.json* is contained in that folder that can be used as a dummy file exported by the HPC with information regarding the submitted jobs. The information contained in the file is based on the format described by the default observers.conf file and can be used for testing purposes.

Prerequisites:

The user that will run the JMA from the start_jma.sh script must be a dedicated for that job user (technical account), since that no-gui client under normal circumstances will run endlessly. That user must be both a system user and an SPDRM user and must fulfill the following requirements:

- Read and write privileges in the JMA directory.
- Read privileges in the no-gui client directory.
- Read privileges for the HPC file with Submitted Jobs' status.
- Read and write privileges in the vault directory.

9.11. JMA Set-up Instruction

The following steps describe the procedure for the set-up of the Job's Monitor Assistant:

1. Stop the SPDRM Server.
2. Edit the "taxis.conf" and add the following key (if not already set):

   ```
   <entry key="noGuiClientPath">/[SPDRM_CLIENT_DIR]/startClient-nogui.sh</entry>
   ```

1. Start the SPDRM server and import the processes "JMA_Process.json" and "HPC_Job_Monitor_sample.json" through SPDRM Client.
2. Edit the script start_jma.sh, located in the jma folder, and set the SPDRM no-gui client directory path at line 3.

3. Edit the “jma.conf” file and change the value of the “path_to_hpc_file” key to the absolute path to the HPC file with the Submitted Jobs’ info. (Do not change the name of the key “*path_to_hpc_file*”)
4. Execute the shell script ‘start_jma.sh’ as *spdrm_jma* user in order to start the Job Monitoring Assistant (JMA) service.

NOTES:

- The jma folder is located in the parent folder of the [SPDRM_CLIENT_DIR] after SPDRM Installation.
- The logs of the JMA are written in a file named "jma.log" that is created in "jma" folder.
- The "JMA_Process" workflow contains a time delay script node that defines the time between two updates of the status of the submitted jobs. The default time is 60 seconds and can be changed by editing its script.
- To stop the Job Monitoring Assistant (JMA) service you should execute the shell script "stop_jma.sh" as *spdrm_jma* user.

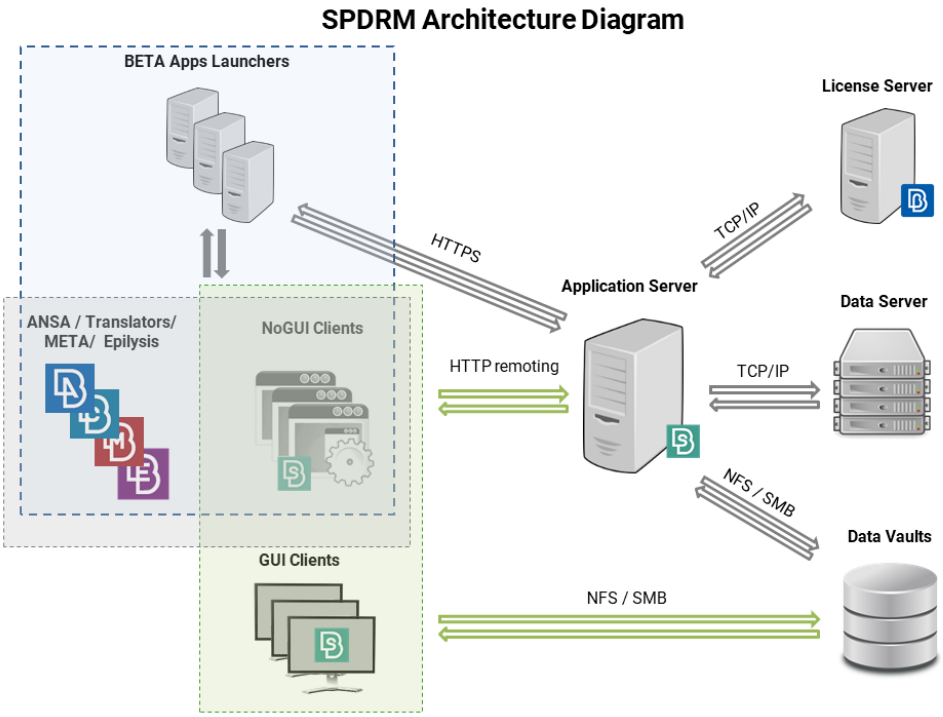
10. BETA Apps Launcher

10.1. Introduction

BETA Apps Launcher is an application that has been designed and implemented in order to give the SPDRM users the opportunity to run resource demanding and time-consuming processes on remote resources. The whole implementation is based on a new light-weight server which runs on the target server machine and communicates with the main SPDRM server in order to open SPDRM NoGUI clients, other BETA Applications or any other application on behalf of SPDRM GUI clients. The aforementioned server is called BETA Apps Launcher.

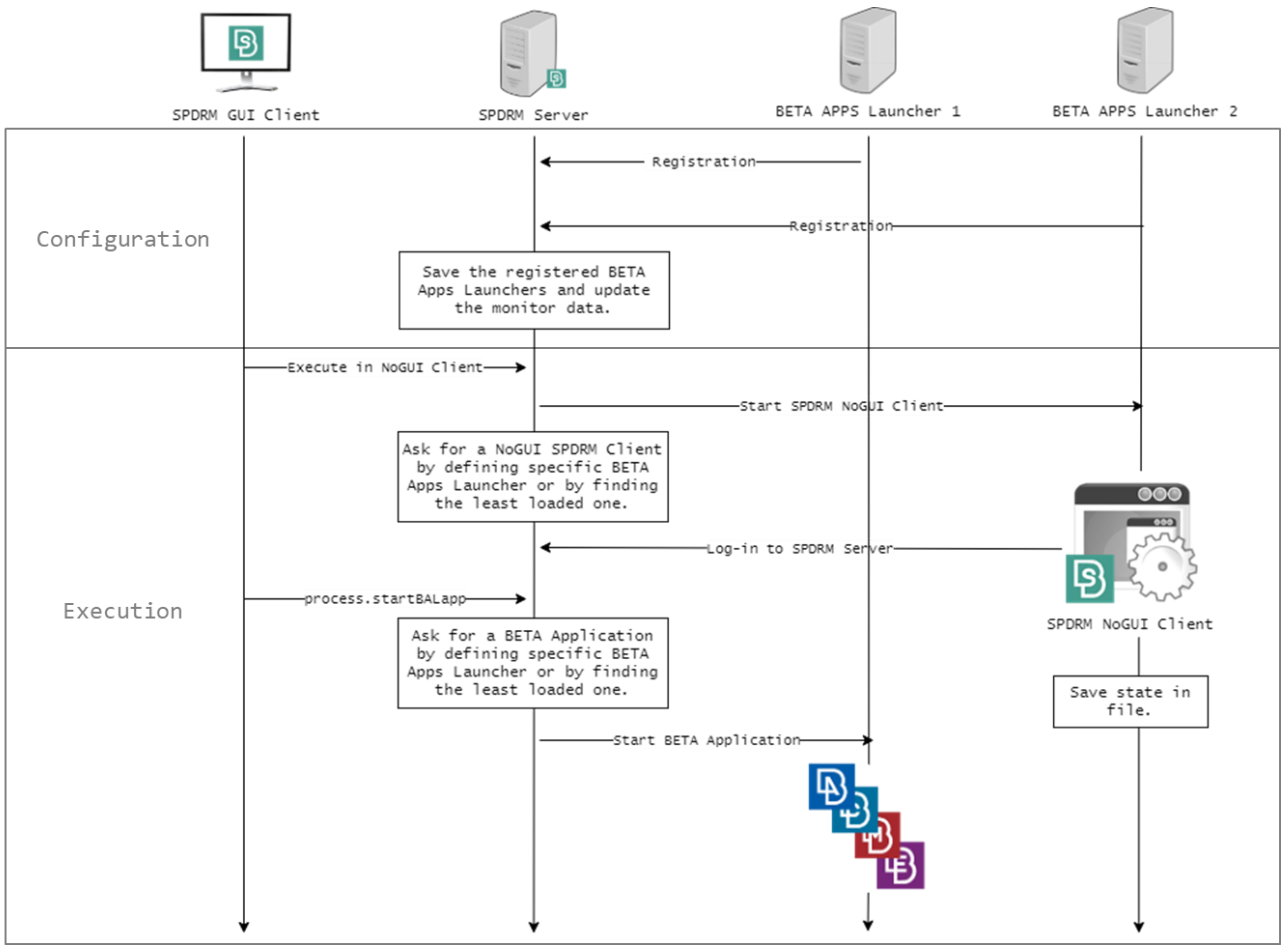
The BETA Apps Launcher comes with embedded queuing, monitoring and emergency mechanisms enabling integrated orchestration of all running applications by the SPDRM server.

The SPDRM architecture -including the BETA Apps Launcher- is explained in the following diagram:



The set-up of the BETA Apps Launcher will be described in the following slides. All files needed can be found in the **betaAppsLauncher** folder in the SPDRM Installation directory.

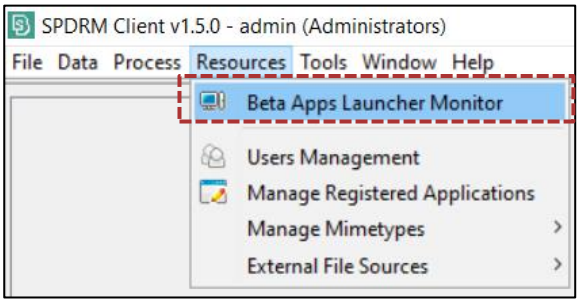
10.2. Architecture Sequence Diagram



10.3. Monitoring

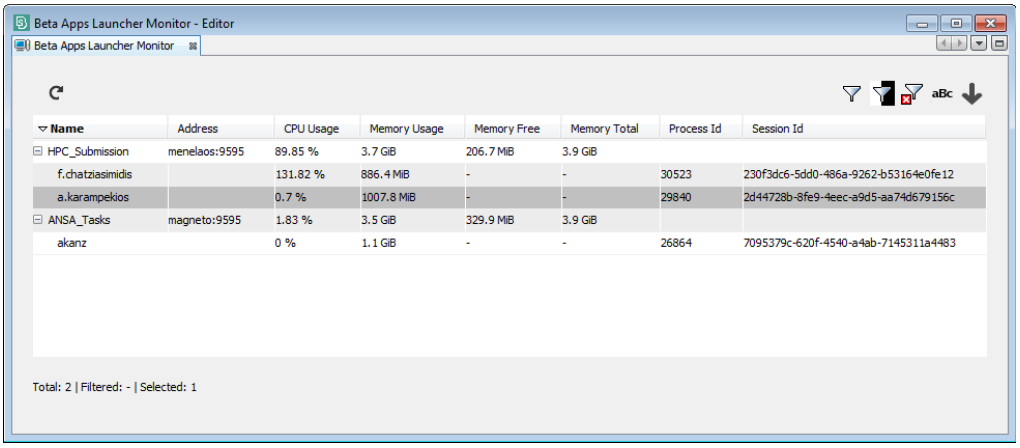
A tool that provides overview of the remote NoGUI Architecture exists that is called **BETA Apps Launcher Monitor**. The BETA Apps Launcher Monitor tool is a window that contains information about the registered BETA Apps Launchers to the SPDRM Environment, as well as useful information about their state and their resources (CPU Usage, Memory Usage, Free Memory and others).

In this window, the administrator can see also the NoGUI Clients that run in each server, alongside some information about each client, such as the logged in user and the client's memory usage.



The BETA Apps Launcher Monitor tool can be accessed through **Resources > BETA Apps Launcher Monitor**.

The image shows the BETA Apps Launcher Monitor, where the registered BETA Apps Launchers are displayed at the top level of the tree and the connected users that run a NoGUI Client can be seen at the expanded tree of each server, alongside with detailed information about the servers' resources.



It is possible to control the BETA Apps Launcher that will be used for the execution of a no-GUI node of a process. Find more information in chapter **Process Design** -> **Assign tasks to No-GUI client using BETA Apps Launcher.**

10.4. Queuing

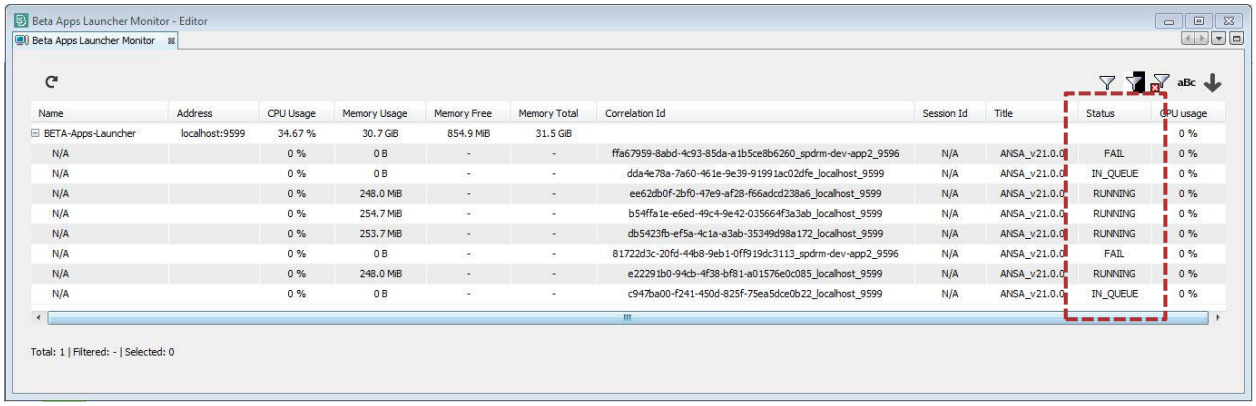
A queuing mechanism is embedded in BETA Apps Launcher and enables handling of system overload during job executions. The queuing mechanism is driven by a scheduling mechanism that controls the executed jobs by the BETA Apps Launcher based on two factors:

- Max number of allowed parallel applications.
- Available resources of the system.

The maximum number of allowed parallel applications can be defined by using the key **parallel-applications-limit** in the BETA Apps Launcher's configuration file, named ***application.properties*** (Detailed description of the configuration file follows in the Setup Guide sub-chapter). The option can be used to prevent irrational use of licenses. If one of the aforementioned rules are violated, the incoming job will not be executed , but it will remain in the queue, until both of the requirements are fulfilled.

Jobs that remain in queue can be found by using the **Beta Apps Launcher Monitor** tool. Under the **Status** field, the current status of each job is displayed. Status can have one of the following values:

1. RUNNING
2. IN_QUEUE
3. FAIL



10.5. Emergency

Utilizing the queuing mechanism, BETA Apps Launcher also provides an emergency mechanism that protects the system from overload or obstruction of other underlying procedures. Emergency mode is activated when a custom rule defined in the configuration file is violated and handles emergency situations following the sequence of events:

- When emergency mode is activated, all running jobs are suspended.
- When resources are regained by the system, BETA Apps Launcher resumes each suspended job one-by-one.
- If a resumed job leads again to an emergency violation, it gets permanently stopped.
- When all resumed jobs that led to emergency have been finished (completed or permanently stopped), then BETA Apps Launcher switches back to normal mode and starts executing jobs from the queue.

The emergency mechanism is able to observe the status of three main system components, **memory** (RAM), **processing unit** (CPU) and **graphics card** (GPU). The emergency mechanism is activated by setting one of the following components in the BETA Apps Launcher's configuration file (*application.properties*):

- **emergency-ram** (e.g. emergency-ram=95 , indicating 95% of available memory limit)
- **emergency-cpu** (e.g. emergency-cpu=90, indicating 90% of processing power limit)
- **emergency-gpu** (e.g. emergency-gpu=85 , indicating 85% of graphics processing power limit)

A fourth related key is available, named **emergency-criteria-operator**, that accepts two values OR and AND. That value defines the relation of the other three resources-related emergency criteria, in order for the emergency mode to be activated. For example, if all three criteria have been defined and OR emergency operator has been set, then emergency mode will be activated if any of the three is violated. If AND emergency operator has been set, emergency mode will be activated only if all three criteria have been violated.

- **emergency-criteria-operator** (e.g. emergency-criteria-operator = AND)

GPU monitoring in emerging mechanism is available only for NVIDIA Graphics Cards.

10.6. Load Balancing Mechanism

Load Balancing mechanism is an embedded mechanism used by SDPRM Server when multiple BETA Apps Launchers have been registered in an SPDRM environment.

When a node is set to be executed in the default BETA Apps Launcher, the Load Balancing mechanism searches for the least loaded registered BETA Apps Launcher in terms of absolute memory (most free memory size).

This mechanism can prevent a job from waiting in queue while another BETA Apps Launcher is available for immediate execution.

10.7. Dynamic Logging

When an application is executed by a BETA Apps Launcher through an SPDRM Client, the executing user may not be able to track the logs of the running application (or even the executing machine of the application), depriving the user of important information regarding the running process.

BETA Apps Launcher provides a dynamic logging mechanism that copies in real-time the application logs in the current working directory in order to be available anytime to the executing user.

10.8. Script API functions for BETA Apps Launcher

A number of script functions are provided through SPDRM Script API that enable handling and monitoring of jobs running in a BETA Apps Launcher through script.

- `process.getAvailableBALs`
- `process.getBALappStatus`
- `process.getBALavailableApps`
- `process.getBALavailableAppsPerType`
- `process.killBALapp`
- `process.pollBALapp`
- `process.startBALapp`

10.9. Setup Guide

The installation of the BETA Apps Launcher consists of the following discrete steps:

1. SSL Certificates creation (Optional)
2. Configuration of the BETA Apps Launcher
3. Configuration of the SPDRM Server
4. Configuration of the Wildfly application server (Conditionally obligatory)
5. Registration of the BETA Apps Launcher to the SPDRM Server

How to enable HTTPS between the SPDRM server and BETA Apps Launcher (Optional)

If no corporate certificates are used for the communication via SSL of two application components, then appropriate certificates must be created following the instructions in the **Setup of RIOC service** 2.2.5 paragraph of this document. If the certificates have been already produced for the RIOC service, the same certificate can be used by the server machine that will host the BETA Apps Launcher so that an HTTPS connection between the SPDRM Server and the BETA Apps Launcher is established properly. If no HTTPS connection is required, the current step can be skipped.

Note: The BETA Apps Launcher package, delivered with the SPDRM Package, contains a pair of key-store and trust-store keys, generated by *BETA CAE Systems* (***spdrm.jks*** and **spdrm-client.jks** respectively). The keys can be used for all SPDRM Services requiring authentication, such as the **BETA Apps Launcher**, the **RIOC service** or the **SPDRM Server – Client SSL communication**. The BETA Apps Launcher package is delivered under the path :

```
<spdrm_installation_folder>/spdrm_installation_wildfly/fileUpdatesDeliverables/betaAppsLauncher.zip
```

Configuration of the BETA Apps Launcher

For the configuration of the BETA Apps Launcher the file called ***application.properties*** must be modified appropriately in the BETA Apps Launcher installation folder and the following properties must be set. An example of such a file can be found in the Appendix.

In the following tables, all available keys for the configuration of a BETA Apps Launcher are described. The first table contains all keys that are obligatory for the registration of a BETA Apps Launcjer. The rest of the tables describe all other optional keys, provided in grouping categories, based on the topic they provide custom configuration. The optional keys can remain unchanged in the sample ***application.properties*** file provided in the installation folder.

Obligatory Keys	Description
`nogui-path`	The path to the SPDRM NoGUI Client (e.g. /path/to/startClient-nogui.sh)
`server.host`	The hostname of the BETA Apps Launcher in order that the SPDRM Server to be able to call it. (e.g. magneto)
`server-name`	The alias of the BETA Apps Launcher that appears in SPDRM Client (e.g. HPC_Submission_Server) (Note: The server-name should not contain spaces)
`server.port`	The port in which the BETA Apps Launcher will start and will be used to communicate with the SPDRM Server. (e.g. 9595)
`spdrm-address`	The SPDRM Server's address in which the BETA Apps Launcher will call it (e.g. http://localhost.localdomain:8080)

Security Keys	Description
`security.require-ssl`	This flag indicates if the BETA Apps Launcher will run in HTTPS mode. Default value is False.
`server.ssl.key-store-type`	The key store type must be JKS.
`server.ssl.key-store`	The server side file where ssl certificates are stored. (e.g. <path/to>/spdrm.jks). Obligatory if key *security.require-ssl = True*.
`server-ssl.key-store-password`	The password of the key-store file. (The one given during keystore export in previous chapter e.g. spdrm123). Obligatory if key *security.require-ssl = True*.
`server.ssl.key-alias`	The alias that has been used for the generation of the key-store. (e.g. spdrm-certificate)
`spdrm-trust-store`	The full path to SPDRM trust-store in case that SPDRM Server runs over HTTPS.
`spdrm-truststore-password`	The password for the SPDRM trust-store in case that SPDRM Server runs over HTTPS.

Logging Keys
`logging.file` *The log file of the BETA Apps Launcher. The user that will start the BETA Apps Launcher must have rights for the creation of such a file. (e.g. ./log/BetaAppsLauncher.log)*
`logging.level.com.betacae.service` *Package debug level. (Options for all logging keys: TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF)(In order of descending detail logging)*
`logging.level.com.betacae.spdrm.commons.os.process` *Package debug level (TRACE, DEBUG, INFO, WARN, ERROR, FATAL or OFF).*
`logging.level.com.betacae.spdrm.nogui.client.manager.httpCommunication` *Package debug level (TRACE, DEBUG, INFO, WARN, ERROR, FATAL or OFF).*
`logging.level.com.betacae.nogui.client.manager.apps` *Package debug level (TRACE, DEBUG, INFO, WARN, ERROR, FATAL or OFF).*
`logging.level.com.betacae.spdrm.common.nogui` *Package debug level (TRACE, DEBUG, INFO, WARN, ERROR, FATAL or OFF).*
`logging.level.org.apache.commons.exec.launcher` *Package debug level (TRACE, DEBUG, INFO, WARN, ERROR, FATAL or OFF).*
`logging.level.org.apache.http` *Package debug level (TRACE, DEBUG, INFO, WARN, ERROR, FATAL or OFF).*
`logging.level.root` *Generic debug level of the packages (TRACE, DEBUG, INFO, WARN, ERROR, FATAL or OFF).*

Applications Setup Keys	Description
`app.applications[i].title`	BETA Apps Launcher can start multiple applications. The applications that can be started, must be pre-defined in the configuration file, by defining 4 information fields for each application and by using a serial number for each app [i]. Here the title (name) of the app must be defined, e.g.: app.applications[0].title=ANSA_v21.0.0
`app.applications[i].path`	Defines the full path to the BETA application's executable file (e.g. /apps/BETA/ansa_v21.0.0/ansa64.sh)
`app.applications[i].version`	Defines the version of the application (e.g. v21.0.0)
`app.applications[i].type`	Defines the type of the application (e.g. META)

General Keys	Description
`jre-path`	The path to the Java Runtime Environment. (That key is necessary only when the BETA Apps Launcher is used for starting any other BETA application apart from SPDRM. When used for starting SPDRM client, the BETA Apps Launcher uses by default the JRE found in SPDRM client folder.
`parallel-applications-limit`	Defines the number of parallel applications that can be executed by the BETA Apps Launcher. Must be an integer and default value is 10.
`pstools-path`	The path to the PsTools suite executable (e.g. *C:\Programs\PsTools*)

root	Defines if the BETA Apps Launcher starts with root privileges. Default value is True.
spring.main.banner-mode	Enables or disables the Spring boot startup banner. Default value is off.

NOTE: For BETA Apps Launcher installation in Windows OS, the download of the PsTools suite is mandatory. The PsTools suite ensures the operation of BAL tools such as Kill, Suspension or Resume of processes. The full path to the PsTools unzipped directory must be defined in the pstools-path key.

10.9.1. Configuration of the SPDRM Server

The following two keys must be added in SPDRM server's configuration file (*taxis.conf)* under the *<system-properties>* component:

```
<entry key="no_gui_mode">remote</entry>
<entry key="noGuiPollTimeoutSeconds">120</entry>
```

Note: The default value of the *noGuiPollTimeoutSeconds* key is 120. Detailed description of those two keys can be found in the SPDRM Server configuration table of paragraph 3.6.2.

10.9.2. Configuration of the Wildfly application server

NOTE: The Configuration of the Wildfly application server step is obligatory if only the BETA Apps Launcher – SPDRM Server communication is over HTTPS (*security.require-ssl = True*).

The following two parameters must be added in Wildfly's configuration file (*standalone-full.xml):*

```
<property name="javax.net.ssl.trustStore" value="/path/to/spdrm-client.jks"/>
<property name="javax.net.ssl.trustStorePassword" value="spdrm123"/>
```

where in the value field, the password for the *spdrm-client.jks* file must be given.

10.9.3. Registration of the BETA Apps Launcher

For the registration of the BETA Apps Launcher follow the instructions below:

1. Create an installation directory (e.g. *BetaAppsLauncher_installation*) in the BETA Apps Launcher.
2. Copy the file *application.properties* in the created directory.
3. Copy the file *beta-apps-launcher-1.0.jar* in the created directory.
4. Make sure that the user that starts the BETA Apps Launcher has write privileges in the directory with the jar file in order that a new file called *noguiClients.json* to be created. This file is used from the BETA Apps Launcher for saving the NoGUI clients which have been started.
5. Run the start script *startBetaAppsLauncher.sh*

Alongside the start*BetaAppsLauncher*.sh script, a script for stopping the BETA Apps Launcher is also provided (stop*BetaAppsLauncher*.sh).

NOTE: In the installation folder of the BETA Apps Launcher, two more useful files can be found:

1. A *BetaAppsLauncher.log* file that contains logs about the server's operation.

2. A *noguiClients.json* file that is a json file where the state of the NoGUI Clients is saved by the BETA Apps Launcher. The BETA Apps Launcher uses that file in order to restore the connection of those clients (if they are still active), in cases of unexpected restarts.

NOTE: The administrator can check the used version of the BETA Apps Launcher anytime in the following link:

`https://<host>:<port>/version`

The link will provide information regarding the **Version**, the **Build date** and the **Changeset** of the registered BETA Apps Launcher, so that the compatibility with the same version of the SPDRM Bundle is ensured.

10.10. Appendix

Example of an *application.properties* file:

```
server.port=9595
server.host=magneto
server-name=HPC_Submission
logging.level.com.betacae.service=DEBUG
logging.file=./log/BetaAppsLauncher.log
security.require-ssl=true
server.ssl.key-store-type=JKS
server.ssl.key-store=./spdrm.jks
server.ssl.key-store-password=spdrm123
server.ssl.key-alias=spdrm-certificate
spdrm-address=http://localhost.localdomain:8080
nogui-path=/path/to/startClient-nogui.sh
```

11. Server-Client SSL communication

11.1. SSL Communication

The current document is a quick guide for enabling SSL communication between the SPDRM server and the SPDRM client. Additionally, the server's configuration described in that document enables the SSL communication of the SPDRM Server with all the applications of the BETA Suite (ANSA/META/KOMVOS).

NOTE: SSL communication between the SPDRM Server and the BETA Suite is provided by version 21.0.x onwards.

Some of the following steps (such as the certificates generation) may need to be skipped, in case of a different procedure followed by each system administrator (e.g. use company's certificates instead).

Whichever the case, the following instructions can be followed to establish SSL configuration from point zero to full configuration.

11.1.1. Generation of certificates

- Open a terminal on the SPDRM server host and change directory to:

  ```
  cd <SPDRM_SERVER_INSTALL_DIR>/server/java/jre_linux/bin/
  ```

- Run the following command to create the server keystore:

  ```
  ./keytool -genkeypair -keystore spdrm.jks -storepass spdrm123 -keypass spdrm123 -keyalg RSA -validity 3650 -alias spdrm-certificate -dname "cn=spdrm-security,o=BetaCAE,c=GR"
  ```

  The above command will create a server keystore named **spdrm.jks** (in current directory) with validity of 10 years, which contains a public/private certificate pair.

- Run the following command to export (in current directory) the server's public certificate named **spdrm-cert.crt**, providing the password defined before (e.g. spdrm123) when asked:

  ```
  ./keytool -export -alias spdrm-certificate -keystore spdrm.jks -rfc -file spdrm-cert.crt
  ```

- Create a client trust-store named **spdrm-client.jks** (in current directory) and store the public certificate there:

  ```
  ./keytool -import -alias spdrm-certificate -file spdrm-cert.crt -keystore spdrm-client.jks
  ```

11.1.2. WildFly configuration

- IMPORTANT: Ensure that the SPDRM server is not running while performing the steps below.
- Copy the client truststore (***spdrm-client.jks***) created earlier in the following directory :

  ```
  <SPDRM_SERVER_INSTALL_DIR>/wildfly/bin/
  ```

- Copy the server truststore (***spdrm.jks***) created earlier in the following directory :

  ```
  <SPDRM_SERVER_INSTALL_DIR>/wildfly/standalone/configuration/
  ```

- Modify the file ***jboss-cli.xml*** located in the following directory :

```
<SPDRM_SERVER_INSTALL_DIR>/wildfly/bin/
```

1. Add the following snippet in the jboss-cli section:

```
<ssl>
     <alias>spdrm-certificate</alias>
     <trust-store>spdrm-client.jks</trust-store>
     <trust-store-password>spdrm123</trust-store-password>
</ssl>
```

   NOTE: The values of the snippet are the respective values of the client trust-store. The **alias** of the certificate as it was imported in the client trust-store, the **trust-store** is the file name of the trust-store and the **trust-store-password** is the password entered when the certificate was imported in the client trust-store.

2. Apply the following changes in the *default-controller* section:
   - The protocol to **https-remoting**
   - The old port to the new https management port (old port + 3) (**e.g.9993**).

- Keep a backup of the current configuration file ***standalone-full.xml*** that is located in:

```
<SPDRM_SERVER_INSTALL_DIR>/wildfly/standalone/configuration/
```

- Modify the file ***standalone-full.xml.ssl*** located in the above directory and update the following snippet for both ManagementRealm and ApplicationRealm security-realm sections:

```
<ssl>

    <keystore path="wildfly.keystore" relative-to="jboss.server.config.dir" key-
    store-password="mypassword" alias="wildfly" key-password="mypassword" gener-
    ate-self-signed-certificate-host="localhost"/>

</ssl>
```

  where:

  - the `wildfly.keystore` should be replaced by `spdrm.jks`
  - the `mypassword` should be replaced by `spdrm123`
  - the `wildfly` should be replaced by `spdrm-certificate`

- Rename the ***standalone-full.xml.ssl*** *to* ***standalone-full.xml*** (overwrite the old one).

- Modify the SPDRM server's configuration file (**taxis.conf**) and add the new *http_port* value, e.g.:

```
<entry key=”http_port”>8443</entry>
```

- Modify the **startServer.sh** script:
  1. Replace the default management port (9990) with the new https management port **9993**.
  2. Replace the default http port (8080) with the new https port **8443**.
  3. Replace in all lines mentioned the string:

```
controller=localhost:${MGNT_PORT}
```

     with the string:

```
controller=https-remoting://localhost:${MGNT_PORT}
```

- Modify the **stopServer.sh** script:
  1. Replace the default management port (9990) with the new https management port **9993**.
  2. Replace the default http port (8080) with the new https port **8443**.

3. Replace in all lines mentioned the string:

   `controller=localhost:${MGNT_PORT}`

   with the string:

   `controller=https-remoting://localhost:${MGNT_PORT}`

- Start the server.

HINT: The WildFly management console can now be accessed in the following address:

`https://<SPDRM_SERVER_HOST>:9993`

NOTE: WildFly interfaces are probably binded to “any address” by default in the configuration file. If the client is not able to connect to the server after the set-up, this one could probably be the source of the problem.

11.1.3. SPDRM Client configuration

- Copy the client trust-store (*spdrm-client.jks*) generated earlier in the SPDRM Client installation directory.
- Modify the ***taxisprops.xml.windows*** and ***taxisprops.xml.linux*** files:

  1. Add the following key:

     `<entry key=”wildfly_protocol”>`**`https-remoting`**`</entry>`

  2. Update the serverPort kay value in order to be compatible with the new server-port.

     `<entry key="serverPort">`**`8443`**`</entry>`

- Modify the **startClient.sh** and **startClient.bat** scripts and add the following arguments in the executable command:

  ```
  -J-Djavax.net.ssl.trustStore=<client_truststore_path> -J-
  Djavax.net.ssl.trustStorePassword=<trustore_password>
  ```

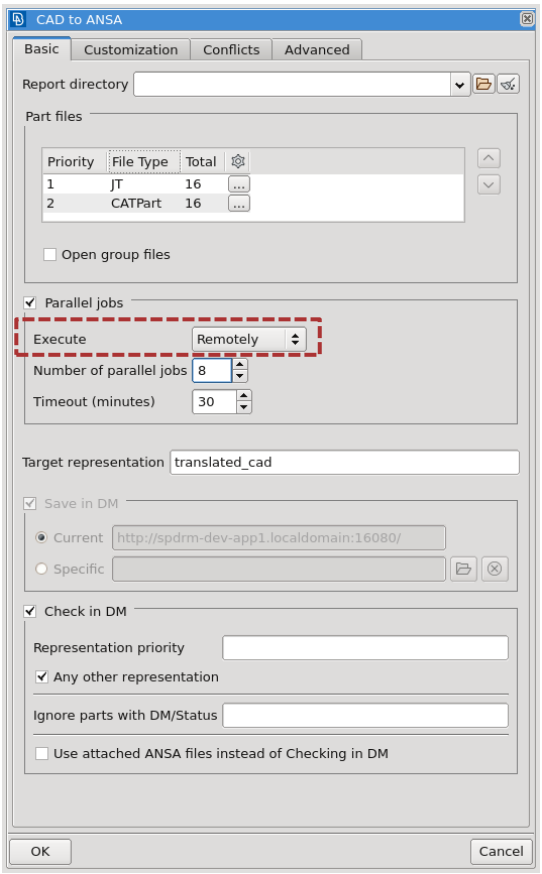
  where:

  - `<client_truststore_path>` must be replaced by the path defined in first step
  - `<trustore_password>` must be replaced by the respective password (e.g. spdrm123).

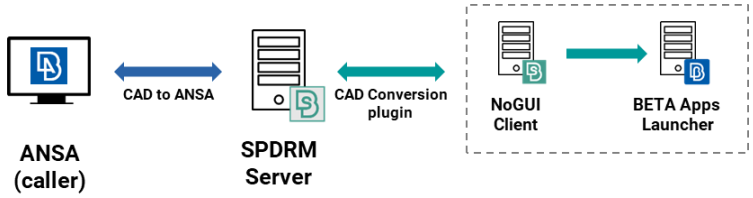
12. Plugins

12.1. CAD Conversion plugin

CAD to ANSA is a tool offered by ANSA to facilitate the translation of CAD files, importing them in the current ANSA model and saving the target representation in the data management repository. The process can be executed sequentially in the existing ANSA session or using parallel ANSA workers for the translation of Parts.



Using **ANSA v22.0.0** or later, the user can enable the execution of *Parallel jobs* and select to execute the process *Remotely*. This requires that the ANSA session is connected to a properly configured environment where SPDRM version **1.5.2** or **1.6.0** (or any other later version) is deployed and the ***CAD Conversion*** SPDRM plugin is installed (as described in the SPDRM Installation Guide). In that case, the translation will be executed on remote resources (i.e. a *BETA Apps Launcher*), utilizing the built-in *CAD Conversion* plugin.

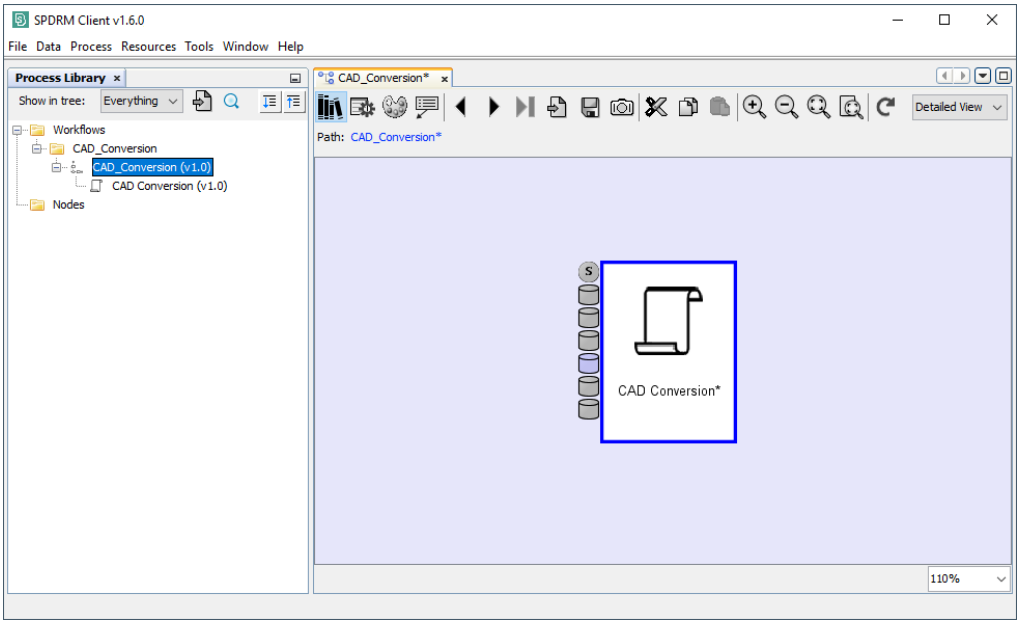


The current ANSA session (i.e. caller) will instantiate the *CAD Conversion* plugin workflow, which will be executed automatically in a NoGUI SPDRM Client. The CAD files will be translated in a remote ANSA session (i.e. worker), in a properly configured BETA Apps launcher, imported back to the model in the ANSA caller and based on the CAD to ANSA settings, saved in the SPDRM data vault with the target representation.

The execution of the plugin is performed automatically in SPDRM, no user action is required.

12.1.1. CAD Conversion plugin workflow

The *CAD Conversion* plugin workflow is available as a template in the Process Library and consists of a single script node.



CAD Conversion node variables

BAL_server *(Type=String, Default value=<empty>)*: A string variable that can be used to explicitly define the BETA Apps launcher to be used by the CAD conversion plugin. Its value needs to be filled with the *Name* of the desired BETA Apps launcher. If empty, the plugin process script will identify an appropriate BETA Apps launcher based on the value of the *BAL_ansa_app_identifier* variable *(see below)*.

BAL_ansa_app_identifier *(Type=String, Default value=<empty>)*: A string variable that can be used to explicitly define the ANSA application in the BETA Apps launcher to be used by the CAD conversion plugin. Its value needs to be filled with the *Title* of the desired application. If empty, the plugin process script will identify an appropriate application based on the running value of the *starting_ansa_version* input slot.

CAD Conversion node Input Slots

starting_ansa_version: A string passed to the workflow from the ANSA caller. It describes the ANSA version in use and can be utilized to find an appropriate application in the BETA Apps launcher.

ansa_defaults: The ANSA defaults file used by the ANSA caller. It will be used by the ANSA worker that will be launched in the BETA Apps launcher to perform the CAD translation

translator_defaults: The translator defaults file used by the ANSA caller. It will be used by the ANSA worker that will be launched in the BETA Apps launcher to perform the CAD translation.

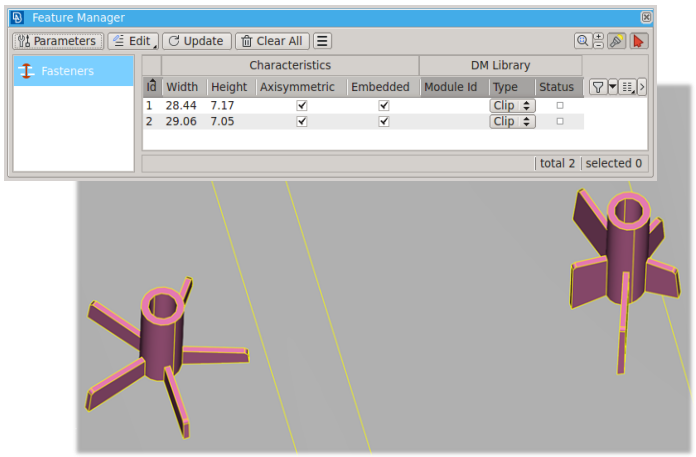
input_data: A json file containing the CAD to ANSA settings from the ANSA caller and the part attributes of the files to be translated

post_actions_script (optional): A python script file that may be defined in the CAD to ANSA settings of the ANSA caller. It will be used by the ANSA worker that will be launched in the BETA Apps launcher to perform the CAD translation, in order to perform a post-translation treatment per part.

ansa_script: A python script file from the SPDRM library, executed by the ANSA worker that will be launched in the BETA Apps launcher to perform the CAD translation.

process_script: A python script file from the SPDRM library that drives the execution of the CAD Conversion SPDRM plugin.

12.2. ML Predictor training



The ***ML Predictor Training*** plugin can be used to train an Embedded Clips Predictor, which can learn to recognize the Clips provided.

It involves the execution of a workflow that takes a training dataset of annotated Clip Feature Entities as input from the end user. Training gets executed in a remote machine with the use of a BETA Apps Launcher.

The output of the workflow is a Predictor DM object that is automatically saved to the DM.

The plugin execution is available using **ANSA** or **KOMVOS v22.0.1** or later, connected to a properly configured environment where **SPDRM v1.6.1** (or any other later version) is deployed and the ***ML Predictor Training*** SPDRM plugin is installed (as described in the SPDRM Installation Guide).

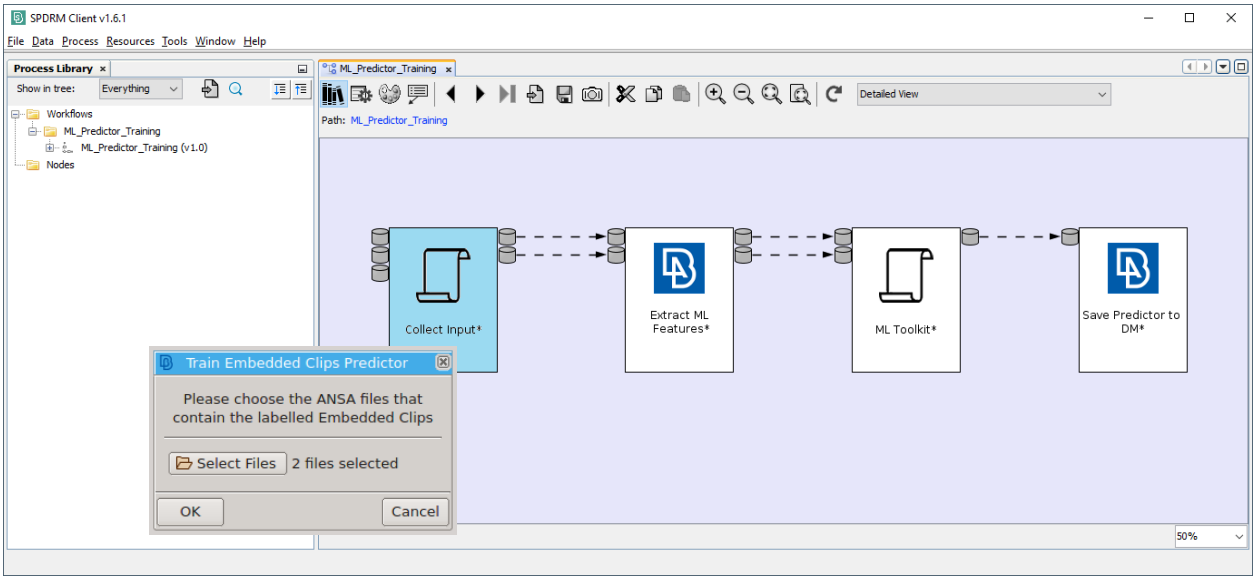
The workflow execution can be triggered through:

- ANSA, using the *DM Browser* action **Script Actions > Train Embedded Clips Predictor (remote)**
- KOMVOS, using the **Machine Learning >Train Embedded Clips Predictor (remote)** action in the *Actions Menu*
- SPDRM, using **Tools > Generic Script Actions > Machine Learning >Train Embedded Clips Predictor (remote)**

Depending on the input dataset size and the GPU memory and compute power of the remote machine, execution can take from several hours to several days.

12.2.1. ML Predictor Training plugin workflow

The *ML Predictor Training* plugin workflow is available as a template in the Process Library and consists of a combination of script and ANSA application script nodes.



All process nodes are automatically executed. Once the training dataset of annotated Clip Feature Entities is provided as input during the execution of the first workflow node, no other user interaction is required through the execution of the plugin.

It should be noted that all process nodes **input** and **output slots** are either scripts loaded from the SPDRM Data Library or data handled internally by the process and **should not be tampered with** by the user.

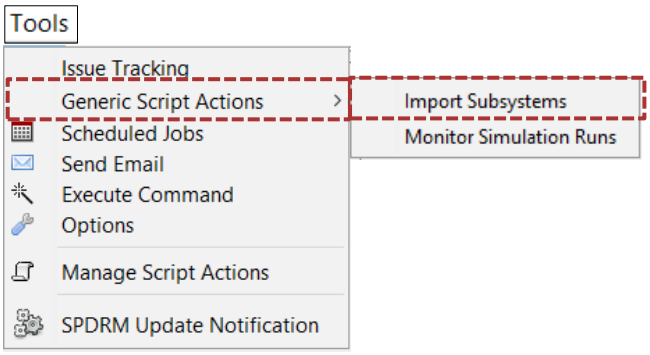
13. Additional topics

13.1. Manage script actions

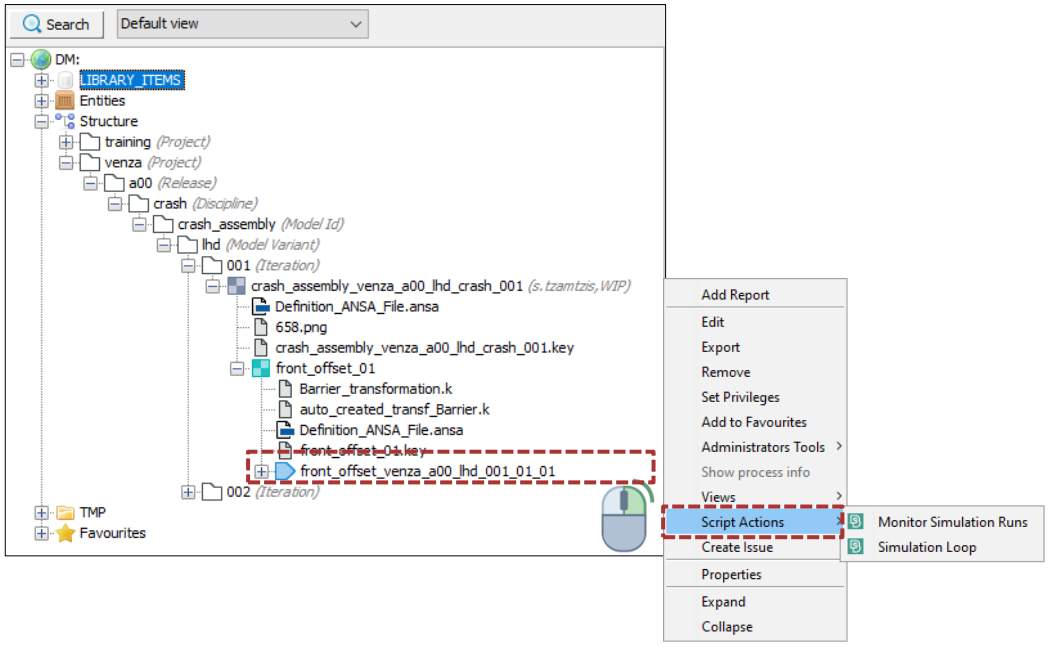
In SPDRM it is possible to perform customized actions on data, called *Script Actions*.

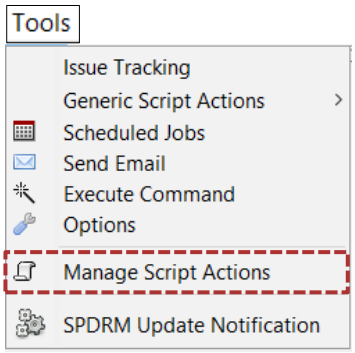
There are two types of Script Actions:

a) Generic Script Actions, which are launched from the SPDRM client menu bar using the option **Tools > Generic Script Actions**



b) DM item-specific Script Actions, which are launched through the **Script Actions** context menu option of DM Items.

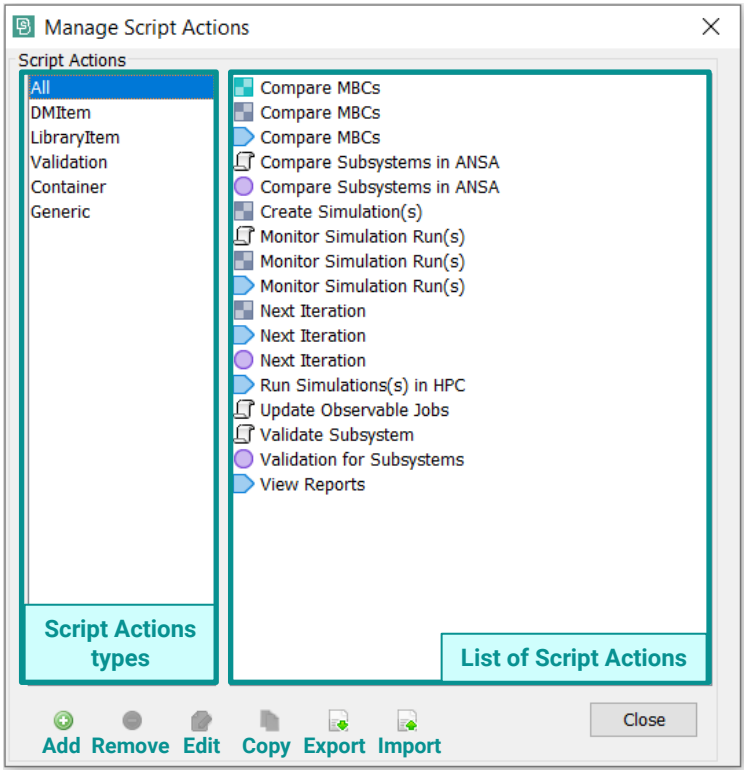




The management of the script actions is performed through the **Tools > Manage Script Actions** option from the SPDRM Client menu bar, which launches the *Manage Script Actions* window.

In the section on the left side of the *Manage Script Actions* window the available Script Action types are listed, while in the section on the right the available Script Actions of the selected type are displayed.

The available Script Action categories are:



All: Lists all available Script Actions, regardless of their type.

Container: Lists Script Actions defined on SPDRM Containers.

DMItem: Lists Script Actions defined on DM items (accessed from the context menu of each DM item).

LibraryItem: Lists Script Actions defined on Library Items (accessed from the context menu of each Rich Library Item).

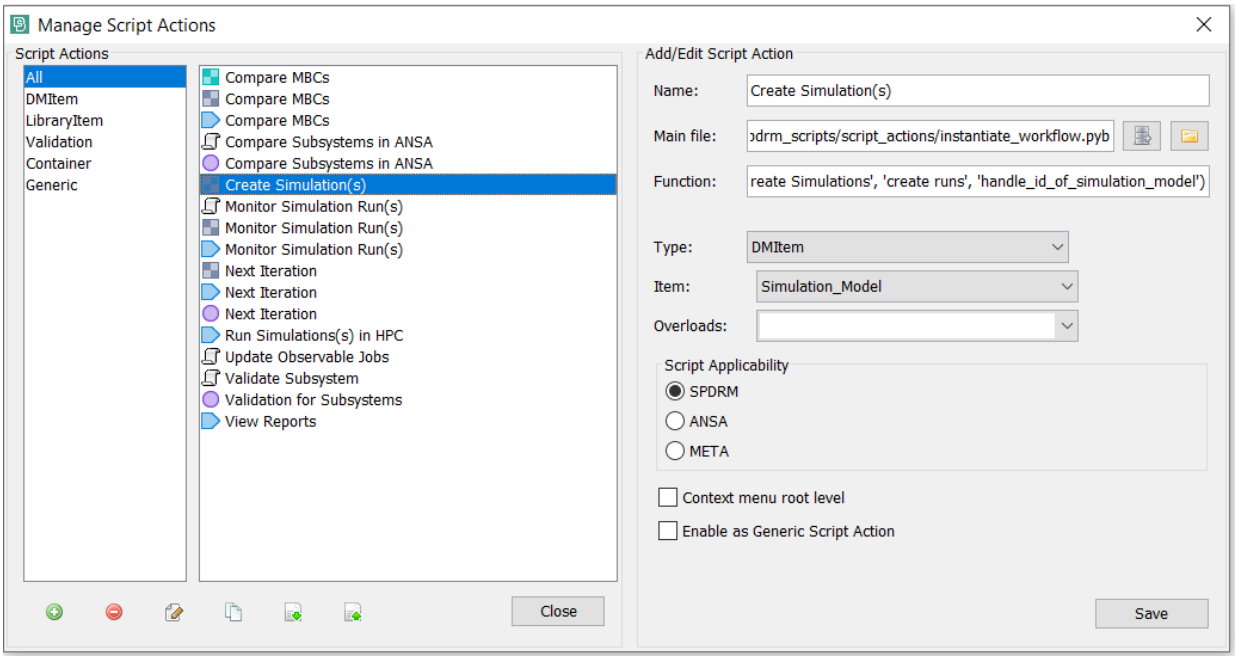
Generic: Lists generic Script Actions (accessed through Tools > Generic Script Actions).

Validation: Lists Script Actions used for the validation of actions on DM items or Rich Library Items.

Using the buttons at the bottom of the *Manage Script Actions* window the user can perform various actions for the management of Script Actions.

Pressing on the **Add** button, the *Add /Edit Script Action* section opens.

The type of the Script Action is the one selected in the list of Script Actions types when the user pressed the **Add** button.



In order to **Add** a new Script Action, the following input is required in the *Add/Edit Script Action* section:

The **Name** of the Script Action, as it will appear in the Generic Script Actions list from the respective option in the SPDRM client menu bar or in the context menu of DM items.

The **Main file** of the script that will be used by the Script Action. The main file can be loaded either from the DM or from the local File System by pressing on the respective button.

The name of the **Function** in the main file that will be executed by the Script Action. For all Script Actions apart from the Validation Actions, the definition of the Function name requires the use of parentheses, while it is also possible to pass arguments to the function.

The Item (DM Item or Rich Library Item) or Container on which the Script Action will be applied, depending on the selected Script Action type being added.

A possible **Overload** of an existing context menu action (e.g. Export, Compare).

The BETA Application on which the script will be executed (SPDRM, ANSA or META).

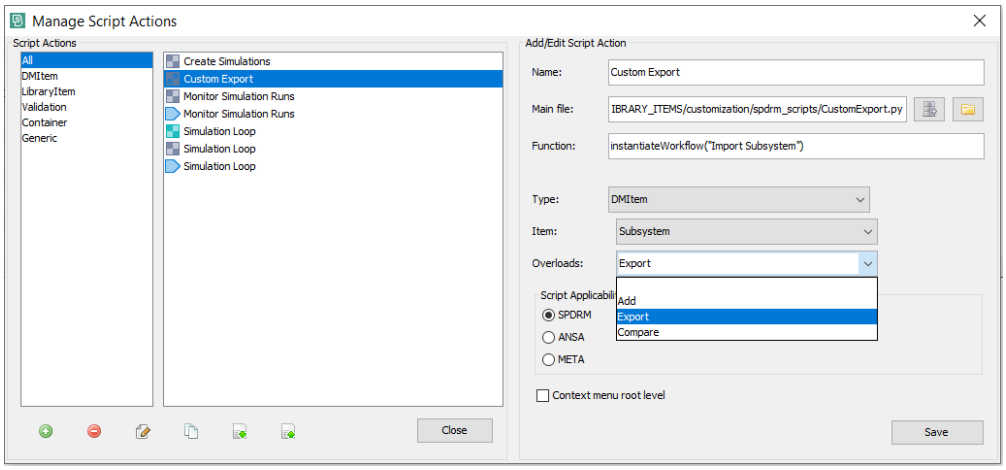
The level of the action in DM Item's context menu. **Context menu root level** checked, sets the action in root level.

Pressing the **Save** button when all settings have been defined, the new Script Action is added.

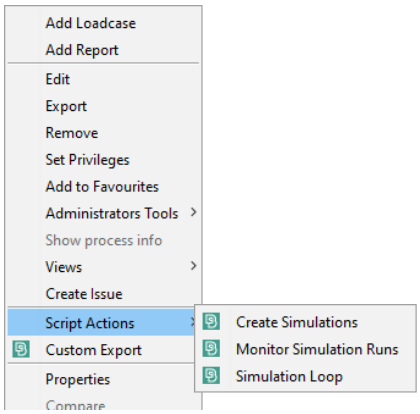
The following options appear only when script action corresponds to type **DM Item** or **LibraryItem**:

Overloads: The override of a built-in action of the context menu can take place by selecting a built-in action to be overloaded by the drop-down menu. The list of the available actions for overload pert type is the following:

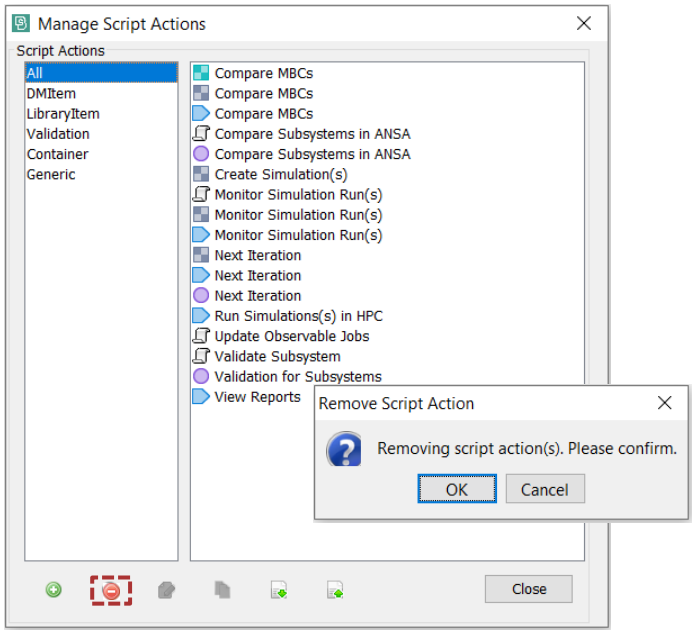
- **DMItem**
  - Export
  - Compare
- **LibraryItem**
  - Export
- **Entity**
  (Part, Subsystem)
  - Add
  - Compare
  - Export



Script Applicability: Defines the target application for the Script Action (SPDRM, ANSA, META). See sub-chapter **Script Actions directly in ANSA/META** in chapter **Process Design** for more information.

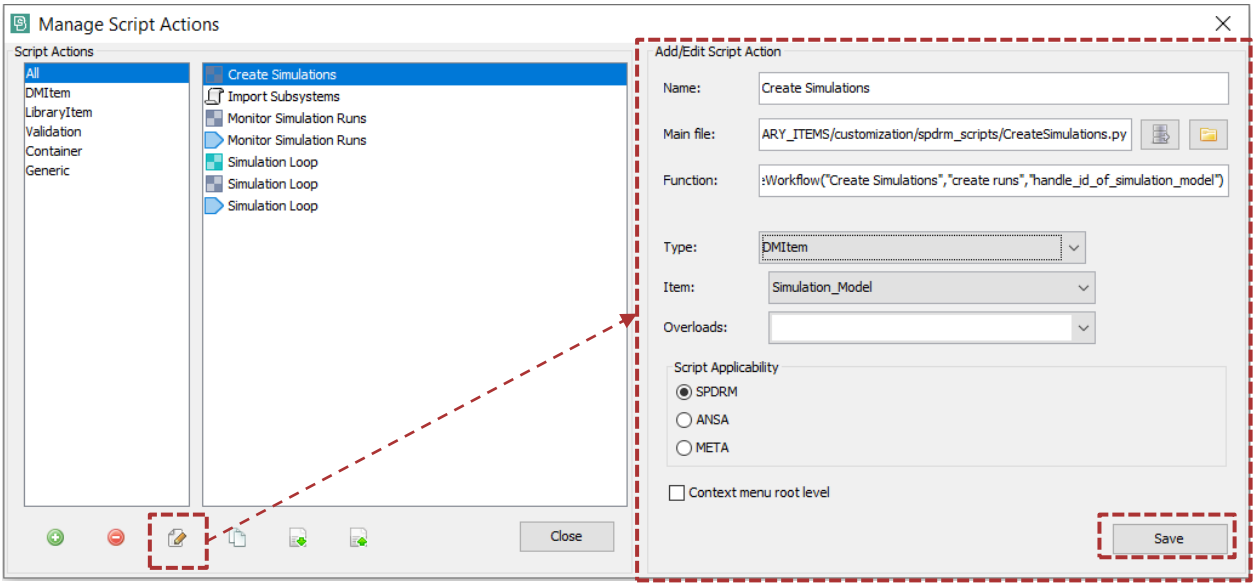


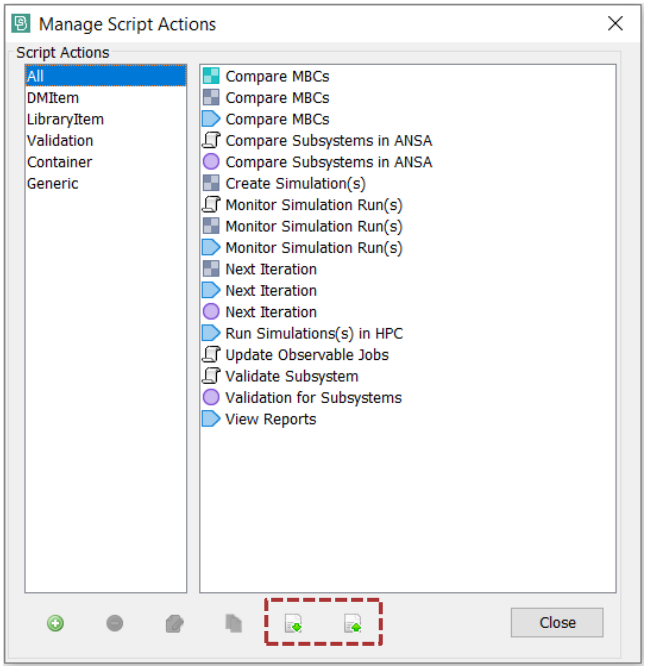
Context menu root level: The set-up of a script action in the root level of the context menu can take place by selecting the checkbox. For example, in this image, **Custom Export** is set in the root level of the context menu, unlike the rest of the Script Actions.



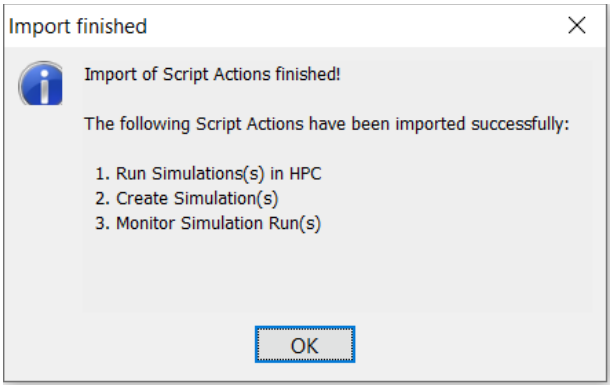
To remove one or more existing Script Actions, the user must first select them in the list of available Script Actions. Pressing on the **Remove** ( ) button, the *Remove Script Action* window opens. The user must confirm or cancel the removal of the Script Action(s) in the *Remove Script Action* window.

In order to **Edit** ( ) an existing Script Action, the user must define its new settings in the *Add/Edit Script Action*. Pressing on the **Edit** button, the Add /Edit Script Action section opens. Once the edited settings have been defined, the user must press **Save**.

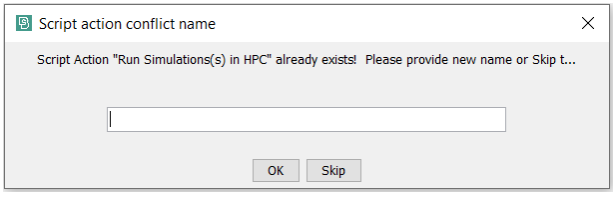




The user can massively **Export Selected** ( ) or **Import** ( ) Script Actions using the respective buttons in the *Manage Script Actions* window. Pressing on the **Export** button, the user has to select a directory on the local File System where the existing Script Actions will be exported as a zipped folder. Once the directory where the Script Action will be exported has been selected, SPDRM exports the zipped folder and a Confirmation Message pops-up.



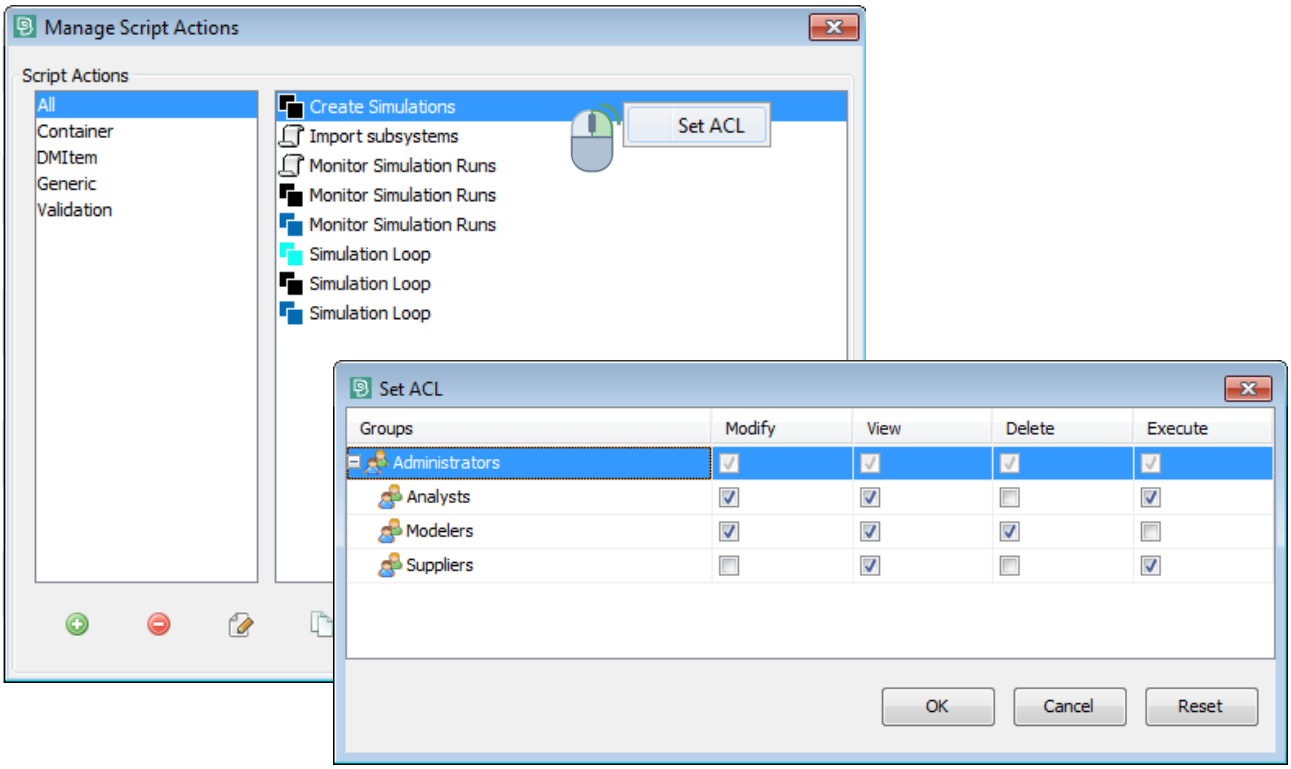
Pressing on the **Import** button, the user has to select the zipped file with the Script Actions that will be imported. If the file being imported contains Script Actions with the same name as an already existing Script Action, the user is prompted to either rename the imported Script Action or skip its import.



Depending on the privileges given, users of specific groups may have access to fewer or none of the available Script Actions, either the Generic Actions from the SPDRM menu bar or the DM item specific Script Actions from the DM item context menu.

Access and modification rights to specific Script Actions can be controlled, through the Access Control List (ACL), by selecting the option **Set ACL** from the context menu in the *Manage Script Actions* window.

The *Set ACL* window appears where the user can define which actions (Modify, View, Delete and Execute) each user group can perform on specific Script Actions.



This feature is particularly useful for both system administrators who want to organize and manage the end user access rights to Script Actions, as well as end users who want to create and use private Script Actions.

13.2. Validation Mechanism

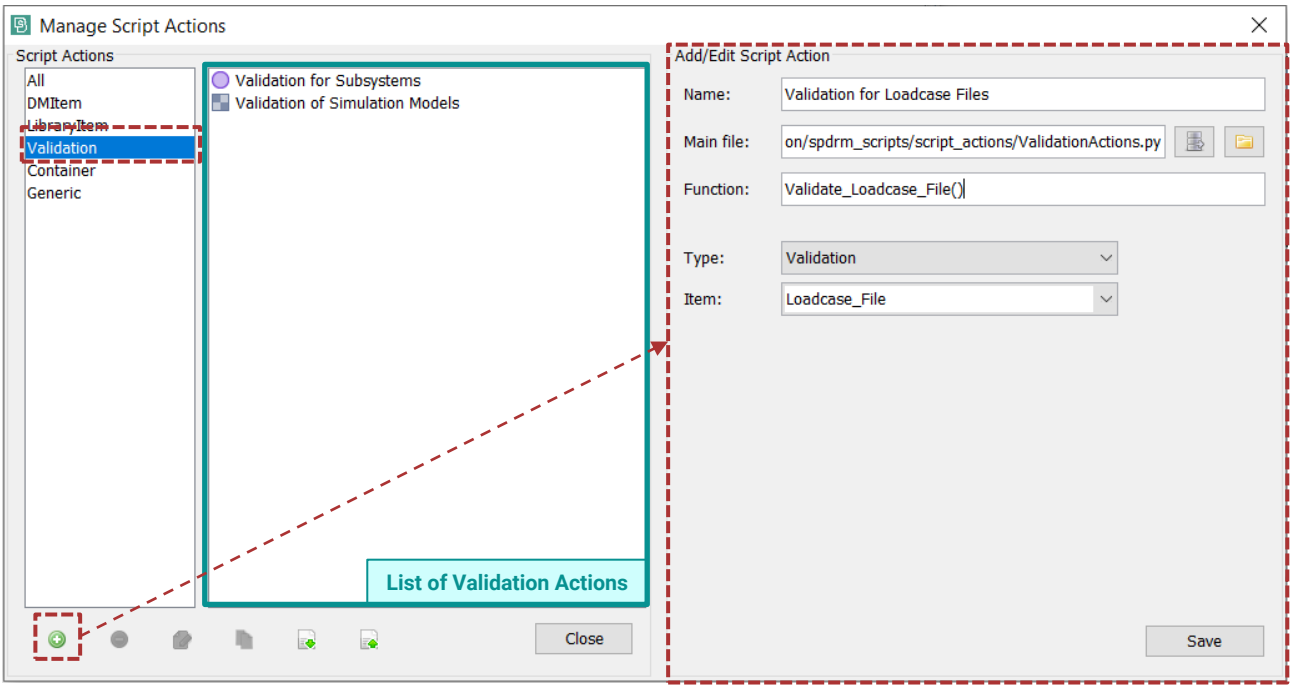
To facilitate the lifecycle management, SPDRM employs a validation mechanism to ensure that any new object complies with the LifeCycle rules, automatically controlling the possible states in the evolution of a Data Object. This is enabled trough the definition of custom Validation Actions on the various Data Types.

The **Main file** is the custom Python script that will be used by the Validation Action. SPDRM currently supports the execution of Validation Actions only with scripts loaded from the DM ( ).

The user must specify the name of the **Function** in the main file that will be executed by the Script Action.

Contrary to other Script Actions, for the definition of the Function name for Validation Actions no parentheses should be used. In addition, no arguments can be passed to this function by the user.

The user must select the **Item** (DM Item or Rich Library Item) for which the Validation Action will be activated.



Contrary to the other Script Action types, Validation Actions cannot be executed on demand by the users. When a Validation Action is defined for a specific Data Type, its execution is triggered automatically by SPDRM during the following operations:

- When a user tries to **Add** a DM object of this specific Data Type, using the respective option in the SPDRM Client.
- When a user tries to **Edit** a DM object of this specific Data Type, using the respective option in the SPDRM Client.
- When the user tries to update a DM object of this specific Data Type, through the **Properties** card in the SPDRM Client.
- In particular for Rich Library Items, when the user tries to **Replace** the file of a DM object of this specific Data Type, or **Add New** version, using the respective options in the SPDRM Client.

- When any of the following SPDRM **script** functions are used in order to create a DM object of this specific Data Type or set its attributes/ additional attributes:
  - `createDMItem()`
  - `createLibraryItem()`
  - `setDMItemAttributes()`
  - `setDMItemAdditionalAttributes()`
  - `setEntityAdditionalAttributes()`
  - `setLibraryItemAttributes()`
  - `setFileAttributes()`
  - `setFileAdditionalAttributes()`

13.2.1. Main script file

The Validation mechanism in SPDRM requires that the function of the main script file defined in the Validaton Action takes a specific argument and returns a specific object.

Validation function argument

When the validation action execution is triggered, SPDRM will pass to the validation script a dictionary as an input argument. This dictionary may contain the following keys:

Key	Key Type	Description
Mode	*String*	the mode of the validation script execution depending on the usage. Possible values: ***Add, Edit, SpinUp, Replace, Properties, Script***
Properties	*Dictionary*	the DM Properties of the DM object
Attributes	*Dictionary*	the DM Attributes of the DM object
Additional Attributes	*Dictionary*	the Additional Attributes of the DM object
DMObject Type	*String*	the Type of the DM object
DMObject Id	*Integer*	(only when SPDRM detects a conflict with an existing object) the Handle Id of the conflicting DM object
Parent Id	*Integer*	the Handle Id of the Parent container of the DM object in the Data Manager Tree
Edited Key	*String*	(only when the validation mode is *Edit)* the Property or Attribute that holds the file that is being edited

Therefore, the definition of the validation script function should have the format:

functionName(argument)

```
def validateRLI(info):
    validation_mode = info['Mode']
    props = info['Properties']
    ...
```

Depending on the Validation Mode and the SPDRM conflict detection outcome, the “DMObject Id”, “Parent Id” and “Edited Key” may not always exist as keys in the validation information PyDictionary.

Validation function return

The validation script should return to SPDRM a validation object that holds information related to the validation mechanism. This is a python object with the following attributes:

Attribute	Attribute Type	Description
is_valid	*Boolean*	**takes the value *True* when then validation is successful or the vale *False* when the validation fails (*Default value: False*)**
create_dm_item	*Boolean*	takes the value ***True*** when SPDRM should create a DM object or the value ***False*** when the validation script will handle the DM object creation (*Default value: True*)
invalid_props	*List*	a list of properties fields that are invalid and should be highlighted in the Add / Properties Card window (applicable when *is_valid=False*)
invalid_attrs	*List*	a list of attributes fields that are invalid and should be highlighted in the Add / Properties Card (applicable when *is_valid=False*)
invalid_addit_attrs	*List*	a list of additional attributes fields that are invalid and should be highlighted in the Add / Properties Card (applicable when *is_valid=False*)
error_message	*String*	a message that will be displayed in a pop-up window, instead of the standard SPDRM error message (applicable when *is_valid=False*)
conflict_resolution	*String*	the versioning policy (overwrite or spin-up) that will be applied for the validated object (applicable when *is_valid=True*). In case of spin-up, the spin-up property must be defined, i.e. Study Version.
modified_files	*Dict*	a dictionary that contains the paths of the files and/or folders that have been modified by the validation script (applicable when *is_valid=True*)

The class that defines the validation object is provided to the validation script by SPDRM and is exposed by a dedicated import in the script. Therefore the validation object must be instantiated in the validation script and at the end of the script execution, returned to SPDRM.

13.2.2. Main script file example

```
# import all SPDRM script library
import SPDRM
# import the SPDRM ValidationResult class
from gr.betacae.clientexecutor.beans import ValidationResult
class LifecycleRules():
    conflict_resolution    =    {'Draft':'OverWrite',    'Validated':'Revision',    'Error':'Revision'}
    RLI_status_transitions   =   {'Draft':['Draft',   'Validated',   'Error'],   'Validated':['Validated', 'Error'], 'Error':['Draft']}
def validateRLI(info):
    # instantiate a validation object from the ValidationResult class
    res = ValidationResult
    # get validation information from the object passed as argument to the validation script
    validation_mode = info['Mode']
    props = info['Properties']
    attrs = info['Attributes']
    additional_attrs= info['Additional Attributes']
    try:
        conflicting_object_id = int(info['DMObject Id']
    except KeyError:
        conflicting_object_id = None
    if validation_mode == 'Properties':
        # the validation script will decide the conflict resolution based on the conflicting RLI Status
        conflicting_props_attrs_info                                                    =
SPDRM.dm.getDMItemPropertiesAndAttributes(conflicting_object_id)
        conflicting_object_status = conflicting_props_attrs_info[2]['Status']
        res.conflict_resolution = LifecycleRules.conflict_resolution

        # the following determine if the Status transition from the conflicting value to the new value is allowed
        accepted_status_values                          =                          Lifecy-
cleRules.RLI_status_transitions[conflicting_object_status]
        if attrs['Status'] not in accepted_status_values:
            res.invalid_attrs.append('Status')
            res.error_message = 'Validator failed due to Status Transition Rule, the new value %s is not allowed!'% attrs['Status']
    # return the validation object back to SPDRM
    return res
```

13.3. Issue Management configuration

Issues in SPDRM are pre-configured with a set of properties, transitions and actions; however their definition can be customized to adapt to the needs of the various engineering teams. Defining additional/alternative status values, user actions and transitions, the built-in workflow can be enhanced or a fully customized issue workflow can be defined.

Through the Issue Management configuration file called *issues.dat*, it is possible to customize several of the issue characteristics, such as:

- The issue properties, as well as the attributes of each property (e.g. the property display name, the accepted values, etc.)

```
properties = project, dueDate, issueDmItem, summary, description, release, pri-
ority, type, rootCause, assignee, status
project.displayName = "Project"
project.acceptedValues = PROJ1, PROJ2, PROJ3, PROJ4, PROJ5, PROJ6, PROJ7, PROJ8
project.defaultValue = PROJ1
```

- The definition of properties as mandatory or optional for the creation of an issue, as well as if the value of each issue property can be edited or not after its creation.

```
project.mandatory = true
project.editable = false
```

- The issue states, the workflow transitions and the user actions that drive each transition.

```
state.new.displayName = "New"
state.assigned.displayName = "Assigned"
action.assign.displayName = "Assign"
action.assign.from = new
action.assign.to = assigned
```

- The issue default and end states and notification actions triggered in state transition:

```
default.state = state
end.states = closed
emailOnStateTransition = true
```

- The necessity for commenting on state change:

```
assignComment.mandatory = true
provideFeedback.mandatory = false
```

- Custom script actions that can be triggered by an issue transition:

```
action.assign.scriptAction = myAction1
scriptAction.myAction1.mainFile = DM:/LIBRARY_ITEMS/scripts/issue_mgmt.py
scriptAction.myAction1.function = doSomething
```

- The date format:

```
date.format = dd/MM/yyyy
```

- The columns that will be displayed in the references table of each issue:

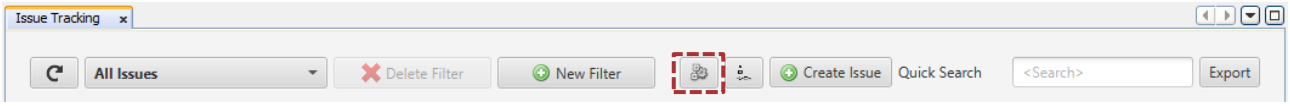
```
dm.columns = Id,Name,Status
```

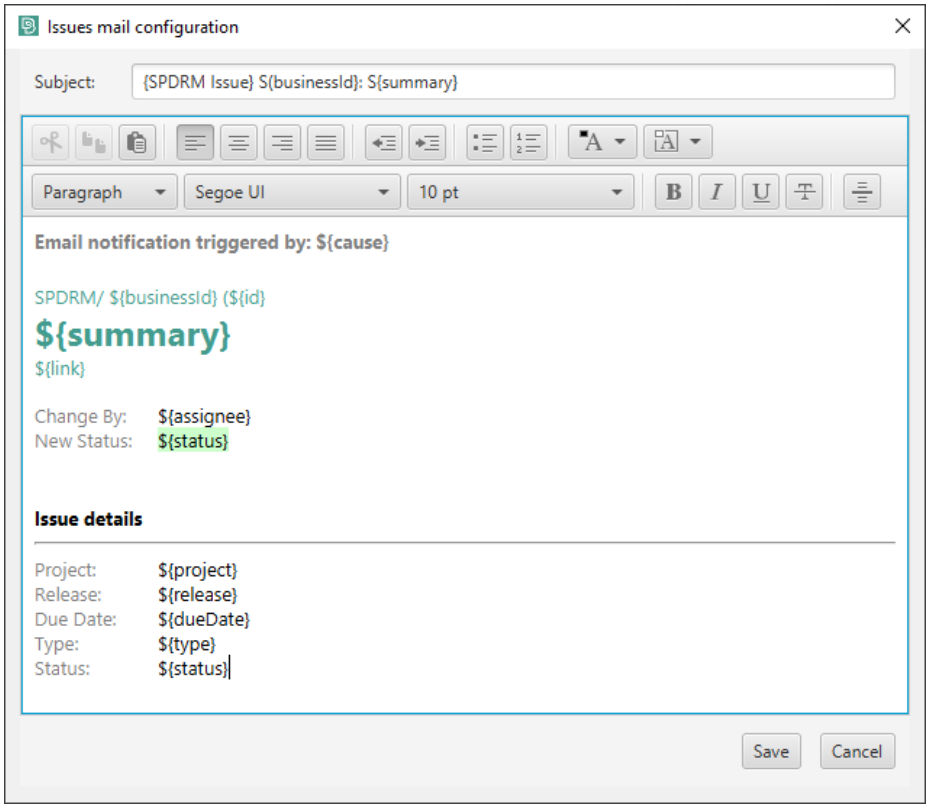
The configuration settings for the Issue Management functionality are included in the ***issues.dat*** file that is located in: `[SPDRM_SERVER_DIR]/wildfly/standalone/configuration/spdrm/.`

13.4. Issue Management email configuration

The SPDRM Issue Management comes with an embedded notification mechanism, ensuring that any user involved with an issue will receive an email when action is required. This email notification content and layout can be customized in order to convey all necessary information in the right context.

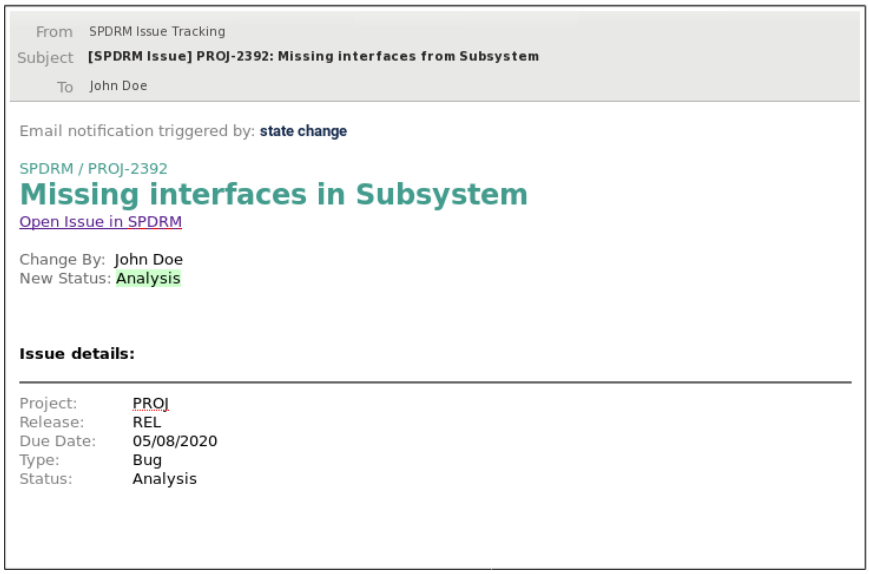
Users with administrator privileges can customize the email notifications using the **View Email Configuration** button.





A concise subject and a body that contains more detailed information about the issue attributes can be defined, as well as the action that triggered the email notification.

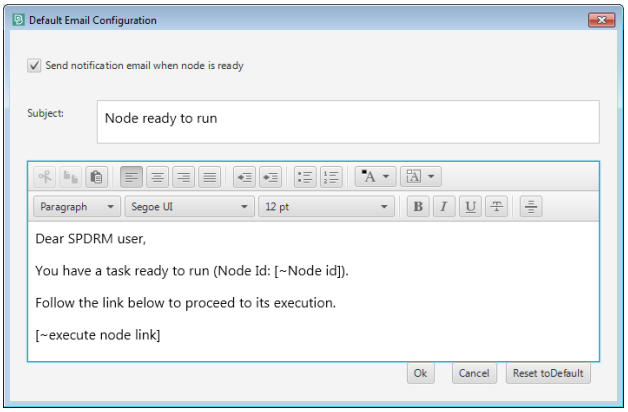
The use of parametric definitions for the issue properties in the email subject and body, as well as html formatting of the email body, is supported.



The email sent by SPDRM contains the desired information, as defined in the issue management email configuration.

13.5. Node email configuration

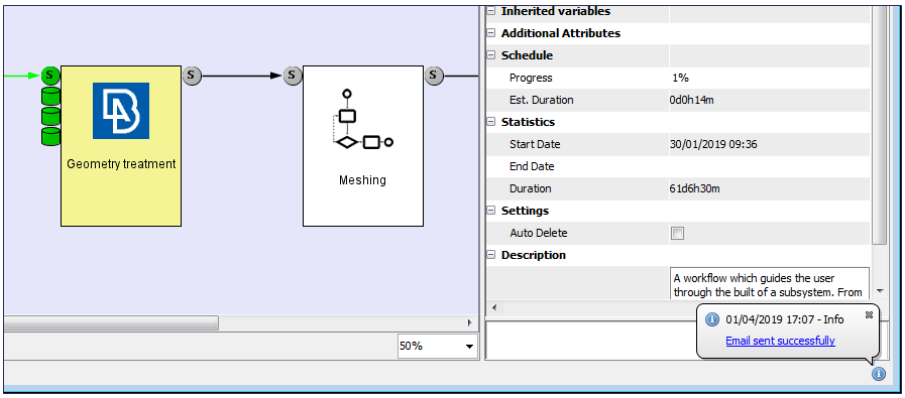
The SPDRM email notification mechanism offers the capability of sending email to the assignees of nodes, when the nodes' state becomes Ready. The mechanism can be generic, sending a pre-defined (template-based) email to the assignee of any node that becomes Ready, or could be node specific.

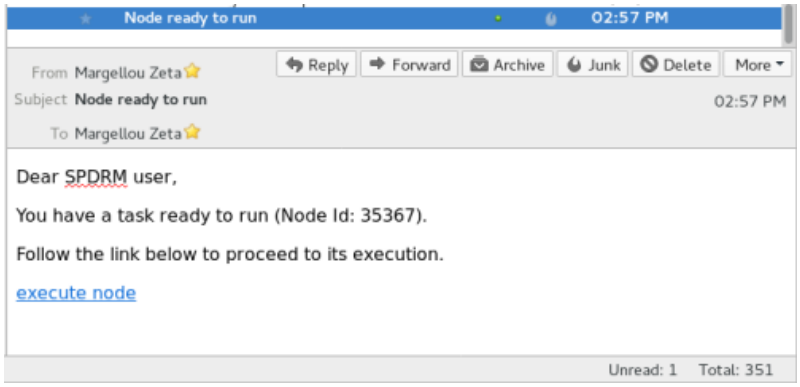


The default email configuration can be defined through the menu bar: **Process > Node Email Configuration**, whereas the node specific email configuration can be set through the **Node Email Configuration** option, in the context menu of each node.

The body of the email can contain parametric information related to the node (e.g. the Node Name, the Node Id, etc.), as well as links to view or execute the node in the SPDRM client.

When the process reaches a task that needs to be executed by some other user and the node email configuration is enabled (either centrally, or on this particular node), the SPDRM sends an email notification to the assignee of this task.





BETA CAE Systems International AG
D4 Business Village Luzern, Platz 4
CH-6039 Root D4, Switzerland
T +41 41 545 3650, F +41 41 545 3651
ansa@beta-cae.com
www.beta-cae.com

physics on screen