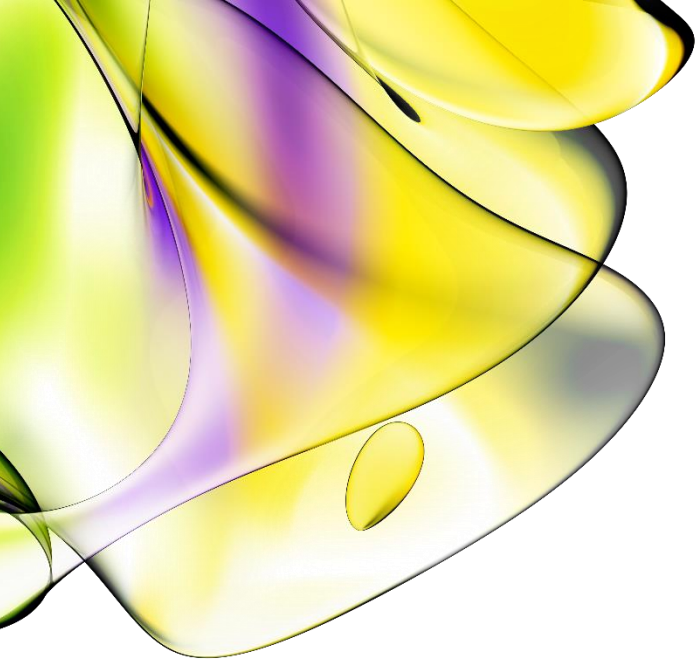


revvity signals

Revvity Signals VitroVivo™ 3.5.0

Installation Guide

Powered by Spotfire®

Last Updated: March 19, 2024

Table of Contents

1. Introduction

Signals VitroVivo™ contains the following components:

- Spotfire® extensions
  - Signals VitroVivo Apps
    - High Throughput and High Content Screening Apps Spotfire® extensions
    - Surface Plasmon Resonance Apps Spotfire® extensions
    - In Vivo Apps Spotfire® extensions (PK Parameters App)
  - Signals Apps Framework Spotfire® extensions
- Images Service
- Calculations Explorer
  - Calculations Explorer Spotfire® extension
- Signals VitroVivo Metastore
  - Signals VitroVivo Metastore service
- Signals Inventa

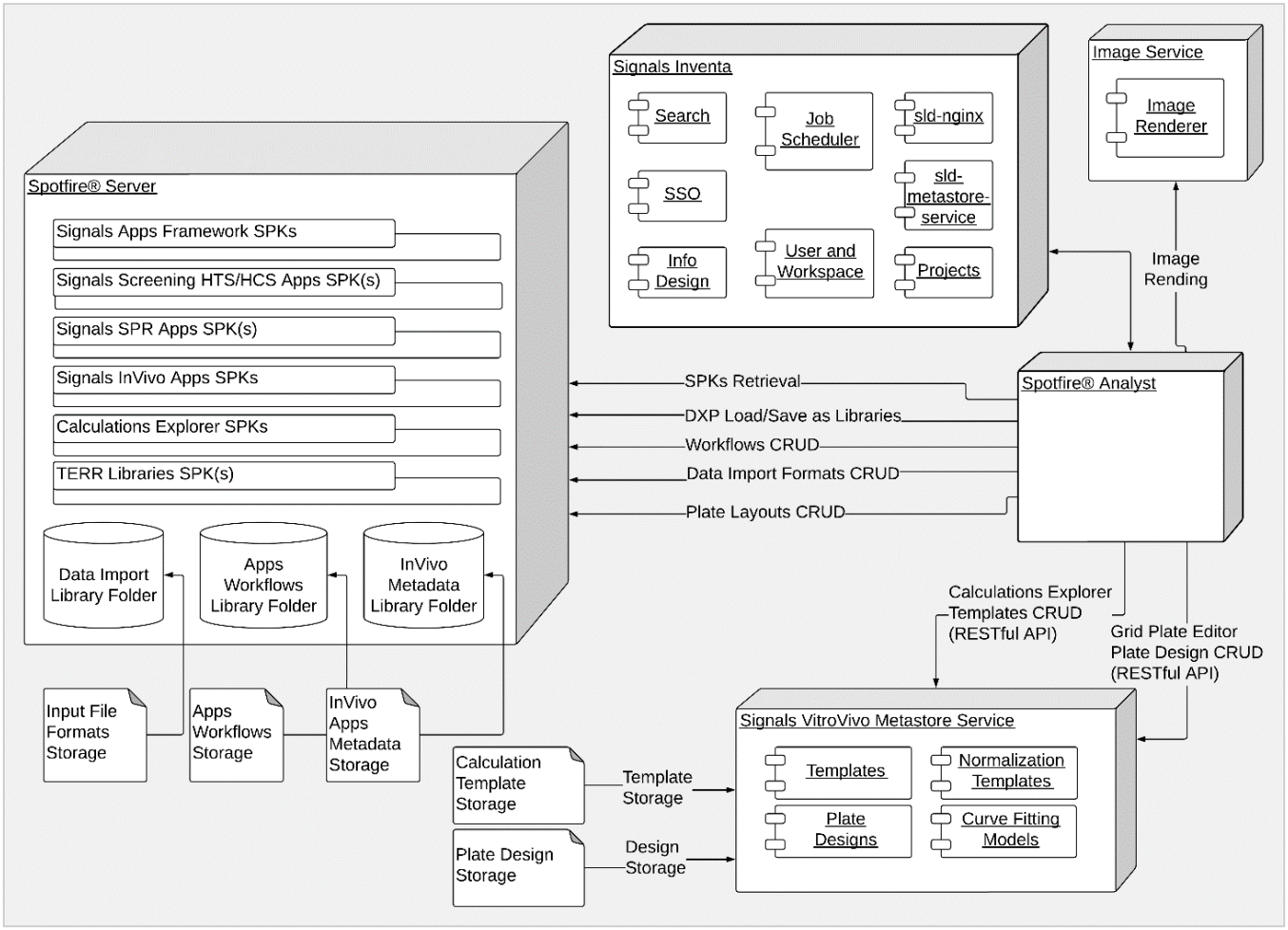


Figure 1-1: *Signals VitroVivo component and deployment diagram*

2. Prerequisites

This Installation Guide explains the steps required to deploy and set up Signals VitroVivo based on the assumption that the prerequisites outlined below have been installed. Refer to the *Revvity Signals VitroVivo System Requirements* document for further details.

1. **Spotfire® Server** and **Analyst Client** have both been installed.
   a. This process is explained in detail in the *Spotfire® Installation and Configuration Manuals* and the *Spotfire® Deployment and Administration Manual.*
   b. You must be a Spotfire® Administrator to perform some of the installation steps described in this guide.
2. The **Calculations Explorer Metastore** service and its backend **MongoDB** require Docker to be deployed. Therefore, a server machine that runs the Docker engine must be set up.
   a. Download and install Docker Compose. For information on Docker Compose refer to https://docs.docker.com/compose/
   b. For more information about the system requirements for a server to run the Docker engine, refer to *Revvity Signals VitroVivo System Requirements*
   c. For more information about the Docker engine installation, refer to the Docker installation guide at https://docs.docker.com/engine/installation/
3. **Signals Inventa,** refer to the *Revvity Signals Inventa Installation Guide* for additional information on this component and its installation.
4. The following software prerequisites are needed in the client where Signals VitroVivo will be used:
   a. Visual C++ 2015-2019 redistributable package (x86)
      i) Download Visual C++ 2015-2019 redistributable package (x86) from Microsoft official site: https://www.microsoft.com/en-GB/download/details.aspx?id=48145
   b. Java 8.0 or higher for 64-bit installed
   c. JAVA_HOME environment variable configured
   d. Microsoft .Net runtime 4.8.x or higher

Note: Ensure the system and web browser regional settings are both configured in English, using dots (.) as the decimal separator [*].

[*] SciStream is supported in systems configured with non-English regional settings so long as dots (.) are used as decimal separators in the input data.

3. Installation

This section explains the installation procedures for Signals VitroVivo. Before proceeding, ensure the installation of the prerequisites described in the previous section have been completed and a valid Spotfire® user license with administration privileges is available.

3.1 Installation Overview

The actual features and functionality of Signals VitroVivo are implemented in software packages. These packages are bundled into a distribution, which can be deployed on the Spotfire® Server as well as some Windows or Linux server systems.

The steps required for installing Signals VitroVivo are:

1. Deployment and configuration of the Signals VitroVivo Apps Spotfire® extension packages [*]
2. Installation and configuration of Signals Inventa (refer to *Revvity Signals Inventa Installation Guide*)
3. License setup of the Signals VitroVivo Apps Spotfire® extensions [*]
4. Configuration of Signals VitroVivo input file formats and Apps Workflow storage [*]
5. Installation of Images Service
6. Deployment of Signals VitroVivo Metastore Service (requires Linux server sudo access if deployed in Linux)
7. Deployment and configuration of the Calculations Explorer Spotfire® extension [*]

[*] Requires Spotfire® administration privileges.

The Signals VitroVivo release goods package includes the following items:

File Name	Description
Signals VitroVivo Apps Spotfire® Distribution Files (SDNs)	
`Signals-VitroVivo.sdn`	The Spotfire® SDN of Signals Apps
Files for Images Service	
`Signals Images Service-<version>.exe`	Images Service installer
Files for Signals VitroVivo Metastore Service	

`docker-compose-linux-<version>.tar.gz`
`docker-compose-linux-ssl-<version>.tar.gz`
`image-<version>.tar.gz`
	The tar.gz file which contains the files necessary to bootstrap the Signals VitroVivo Metastore includes:
	1. .env: The directory-level environment variable settings file, this file will be loaded when docker-compose starts the services defined within the docker-compose.yml file
	2. docker-compose.yml: The YAML configuration file for the docker-compose command line to start the Docker containers for Signals VitroVivo Metastore Services

Table 3-1: *List of items within Signals VitroVivo Release Goods*

Note: Remember to substitute the `<version>` with the appropriate value according to the file name of the package during the following installation and deployment process.

Note: The goods for the installation of Signals Inventa are provided separately as described in the *Revvity Signals Inventa Installation Guide*.

3.2 Signals VitroVivo Apps Spotfire® Extension Packages Installation and Configuration

3.2.1 Deploying Spotfire® Extension Packages

Signals Apps are delivered as a set of SPK package files or in a single SDN distribution file. These files are deployed to the Spotfire® Server to enable the *Signals Apps Tool* which opens the *Signals Apps* catalog.

Deploy in the Spotfire® server area the SDN listed under the section **Signals VitroVivo Apps Spotfire® distribution files (SDNs)** of Table 3-1.

After deployment the following packages will appear in the respective area:

- Calculations Explorer <*version*>
- Image Discovery <*version*>
- Image Import <*version*>
- Integromics.Hcs <*version*>
- Integromics.Hcs.Help <*version*>
- Integromics.Hcs.Python <*version*>
- Integromics.Hcs.ThirdParty <*version*>
- Integromics.Hcs.Web <*version*>
- Integromics.Python.Executor <*version*>
- Integromics.Python.Runtime.2.7 <*version*>
- SciStream <*version*>
- Signals Analytics Apps for Screening <*version*>
- Signals Analytics Historical Data App <*version*>
- Signals Analytics Signals Data Import App <*version*>
- Signals Apps <*version*>
- Signals Apps Groups Panel <*version*>
- Signals Notebook Spotfire Opener <*version*>
- Signals Screening Common <*version*>
- Signals SPR Alignment App <*version*>
- Signals SPR Blank Subtraction App <*version*>
- Signals SPR Common <*version*>
- Signals SPR Cropping App <*version*>
- Signals SPR Data Import App <*version*>
- Signals SPR Data Readers <*version*>
- Signals SPR Export Report App <*version*>
- Signals SPR Hit Selection App <*version*>
- Signals SPR MultiCycle Kinetic Analysis App <*version*>
- Signals SPR RAC App <*version*>
- Signals SPR Reference App <*version*>
- Signals SPR Relative Potency App <*version*>
- Signals SPR ReportPoints App <*version*>
- Signals SPR Single Cycle Kinetics App <*version*>
- Signals SPR Solvent Correction App <*version*>

- Signals SPR Steady State Analysis App <*version*>
- Signals SPR TraceDrawer Export App <*version*>
- Signals SPR Zeroing App <*version*>
- Signals VitroVivo Product <*version*>
- Signals.InVivo.Common <*version*>
- Signals.InVivo.PKParameters <*version*>
- Signals.Utils.Metastore <*version*>

Note: `<version>` corresponds to the current value of the packages' version during the deployment process related with the Signals VitroVivo Apps' version. It can be confirmed in the **Version** column in the deployment area information.
Note: For additional details on deploying new packages and distributions, please refer to the *Spotfire® Server and Environment Installation and Administration*.

3.2.2 Setting up Spotfire® Extension Licenses

Some features are protected by a license to allow administrators to control user access. Administrators are granted access to all Apps, even those with custom licenses. Users have the same access level as the group they belong to.

All Signals VitroVivo user groups must have certain license functions enabled to use the functionality. To grant group access to a licensed App, the administrator can configure license from the Spotfire® Administration Manager.

Follow Spotfire's® instructions on how to edit group license rights. Both the Apps and features licenses are categorized as Spotfire® Extension licenses.

Enable the following licenses for every user group with access to Signals VitroVivo:

- Access to Extensions
- Columbus Navigator for Spotfire®
- High Content Profiler License
- High Content Profiler Panel License
- SciStream API Feature
- SciStream Data Source
- Image Discovery
- Author Scripts
- Signals Apps Dependency Manager
- Signals Apps Tool

Enable the following licenses on the required groups:

- Signals Apps Workflows Role – Admin: For those groups that will administer the Apps Workflows
- Signals Apps Workflows Role – Author: For those groups that will create Apps Workflows

Disable the following licenses for every user group with access to Signals VitroVivo:

- Signals Apps Data Model Editor
- Signals Apps Development

In addition to the licenses specific to the VitroVivo solution, there are certain Spotfire® licenses that need to be active for all App functionalities to be available. These are:

- Spotfire® Consumer
- Spotfire® Enterprise Player
- Spotfire® Analyst
- Spotfire® Business Author
- Spotfire® Connectors (if any connectors are needed for the Historical Data App)
- Spotfire® Advanced Analytics
- Spotfire® Information Modeler

3.2.3 Signals VitroVivo Apps Spotfire® Extension Configuration

3.2.3.1 Configuring the Signals VitroVivo Input Data Format Storage for the Data Import App

To share input file formats across users, Signals VitroVivo requires that a library folder is configured on the server and the library folder path is defined in the corresponding Signals VitroVivo preference for every Spotfire® user group with access to Signals VitroVivo.

1. Open Spotfire® as an administrator and go to **Tools > Library administration** and create a new library folder at the Spotfire® Server library to store the input data formats.
2. Go to **Permissions** for the selected folder and give *Browse+Access+Modify permissions* to the folder defined in step 1 to the Signals VitroVivo Spotfire® user groups.
3. Go to **Tools > Administration manager > Preferences** and select the desired user groups to add to the library folder and initialize the **Signals Apps > Data Import > Library Folder** preference with the folder path created in step 1.
4. To configure different folders for different user groups, repeat steps 1 through 3 for each user group.

3.2.3.2 Configuring the Signals VitroVivo Data Table Storage for the Editable Data Grid App

To share data tables for the Editable Data Grid App across users, Signals VitroVivo requires that a library folder is configured on the server and the library folder path is defined in the corresponding Signals VitroVivo preference for every Spotfire® user group with access to Signals VitroVivo.

1. Open Spotfire® as an administrator and go to **Tools > Library administration** and create a new library folder at the Spotfire® Server library to store the data tables.
2. Go to **Permissions** for the selected folder and give *Browse+Access+Modify permissions* to the folder defined in step 1 to the Signals VitroVivo Spotfire® user groups.
3. Go to **Tools > Administration manager > Preferences** and select the user groups that you want to add to the library folder and initialize the **Signals Apps > Editable Data Grid > Library Folder** preference with the folder path created in step 1.
4. To configure different folders for different user groups, repeat steps 1 through 3 for each user group.

3.2.3.3 Configuring the Signals VitroVivo Metastore for the Grid Plate Editor App

To function correctly, the **Grid Plate Editor** requires a connection to the Signals VitroVivo Metastore (see Signals VitroVivo Metastore Service) where the URL to the Metastore server must be defined in the corresponding Signals VitroVivo preference for every Spotfire® user group using the Grid Plate Editor. To do this:

1. Deploy the Signals VitroVivo Metastore as described in this manual. Alternatively, use an already running instance of the Signals VitroVivo Metastore.
2. Open Spotfire® as an administrator and go to **Tools > Administration manager**.
3. In the **Preferences** tab of the Administration manager, select the user groups that should have access to the Signals VitroVivo Metastore and add the URL of the Signals VitroVivo Metastore that should be used in the **Signals VitroVivo Metastore > Service URL** preference.

3.2.3.4 Configuring Signals VitroVivo App Workflows Storage

To share Signals VitroVivo App Workflows across users, Signals VitroVivo requires that a library folder is configured on the server and the library folder path is defined in the corresponding Signals VitroVivo preference for every Spotfire® user group with access to Signals VitroVivo. To share the data models within a Spotfire® user group, configure the data model storage for the Spotfire® user group:

1. Open Spotfire® Analyst as an administrator.
2. Create a new folder to store the Workflows for a given user group.
3. Grant *Browse+Access+Modify* permissions to the folder for the group.
4. Open **Tools > Administration manager > Preferences**. Set the **Signals Apps > Settings > Protocol Library Folder** preference to the path to the previously created folder.

Note: When adding the library path do not include “Library/”, as all paths are defined relative to this folder.

Note: Changes will be available when the user logs in again.

3.2.3.5 Configuring Signals VitroVivo Apps Notebook Connection

The administrator can configure the Signals Notebook server and the Signals Notebook experiment for a specific Spotfire® user group:

1. As an administrator, navigate to **Tools > Administration manager > Preferences** and select the user group(s) to configure the Signals Notebook server and experiment.
2. Navigate to **Signals Notebook > Signals Notebook Settings > Signals Notebook url** and add the url of the desired Notebook.
3. From the same menu, select **Signals Notebook experiment** and add the experiment ID of the desired Notebook experiment. If not provided, the experiment used by the document will be set by the user or by the system if the DXP was opened from Signals Notebook directly).

Note: Changes will not be available until users logs in again.

If the user sets the **Signals Notebook url** preference, the set server url form will not appear. If the user sets the **Signals Notebook experiment** preference the select experiment form will not appear during the Signals Notebook server connection process.

3.2.3.6 Configuring Signals Data Factory Connection for the Signals Data Import App

To retrieve data from an existing Signals Data Factory from the Spotfire® Apps, Signals VitroVivo requires the correct preferences to be configured in Spotfire®. To allow this for a specific user group, configure the Signals Lead Discovery preferences for the Spotfire® user group:

1. Navigate to **Tools > Administration manager > Preferences** and select the user groups to allow connection to the Signals Data Factory.
2. Navigate to **Signals Lead Discovery > General > SDF Base URL** and add the url of the desired Signals Data Factory.

Note: When configuring the connection to SDF, both the Spotfire® connection and SDF connection should follow the same security configuration, https connection for one and http for the other is not supported.

3.2.4 Signals VitroVivo Metastore Service

There are two flavors to use, according to whether SSL is desired. These are the provided files:

- `docker-compose-linux-<version>.tar.gz`
- `docker-compose-linux-ssl-<version>.tar.gz`

Note: Support for Windows server has been deprecated.

3.2.4.1 Deploying the Signals VitroVivo Metastore Service on a Linux Server Machine (no-SSL)

Follow the steps below to deploy the Signals VitroVivo Metastore Service with docker-compose on a Linux Machine:

1. Log on to the Linux server machine that runs the Docker engine and docker-compose with your regular account.
2. Install Docker following the instructions available at https://docs.docker.com/engine/install/ubuntu/.
   If using ubuntu 20.04, some steps may be performed differently. A description of these steps is available in the Troubleshooting section.
3. Install Docker compose. Follow the instructions at https://docs.docker.com/compose/install/. Deploying the Signals VitroVivo Metastore Service with docker-compose requires docker-compose version 1.15.0 or higher to be installed on the same machine as the one that contains the docker engine.
4. Extract the `docker-compose-linux-<version>.tar.gz` in the desired location in the file system by using the following command:

   ```
   sudo tar xzvf docker-compose-linux-<version>.tar.gz -C <path_to_extract>
   ```

   Where `<path_to_extract>` is the file system location to extract the files.

   Remember to replace the `<version>` with the actual value from the file name.

5. Edit `.env` file with a text editor, modifying the environment variable settings based on the existing setup. Note this file is by default hidden when using the “ls” command, to see it use “ls -a”:
   - `MONGODB_DATABASE_VOLUME`: A string which represents the docker volume mapping between the file system paths from the host Linux machine and the docker container. In this case it specifies the MongoDB

database file path mapping. This string contains three parts separated by colons. The first part is the file system path to store MongoDB database files on your host machine, the second is the path in the docker container where MongoDB will save the data, and the third is the mapping option. You only need to change the first part to match the existing setup.

- `METASTORE_PORT`: The TCP/IP port number for the Signals VitroVivo Metastore Service.

6. Under the same directory where the above file exists, execute the following shell command to load the latest metastore image into your local docker:

```
sudo docker load -i image-<version>.tar.gz
```

7. Then, within the same directory, execute the following shell command to start the Signals VitroVivo Metastore Service as a daemon process:

```
sudo docker-compose up -d
```

8. To verify that the service Docker containers have started successfully, use a web browser to access the service endpoint below to see if it returns the version number of the deployed service:

```
http://<server_name>:<port>/
```

Where the `<server_name>` is the name of the server which hosts the Signals VitroVivo Metastore Docker container, the `<port>` represents the Signals VitroVivo Metastore Service port number specified in the `.env` file.

Verify the deployment by executing the following command under Linux shell:

```
curl http://<server_name>:<port>/calculationProtocols && echo
```

Or by executing the following command under Windows PowerShell 6.0 or above:

```
Invoke-RestMethod http://<server_name>:<port>/calculationProtocols
```

Where the `<server_name>` is the name of the server which hosts the Signals VitroVivo Metastore Docker container, the `<port>` represents the Signals VitroVivo Metastore Service port number specified in the `.env` file.

3.2.4.2 Deploying the Signals VitroVivo Metastore Service on a Linux Machine (SSL)

Follow the steps below to deploy the Signals VitroVivo Metastore Service with docker-compose on a Linux Machine:

1. Log on to the Linux server machine that runs the Docker engine and docker-compose with your regular account.
2. Install Docker following the instructions available at https://docs.docker.com/engine/install/ubuntu/.

   If using ubuntu 20.04, some steps may be performed differently. A description of these steps is available in the Troubleshooting section.
3. Install Docker compose. Follow the instructions at https://docs.docker.com/compose/install/. Deploying the Signals VitroVivo Metastore Service with docker-compose requires docker-compose version 1.15.0 or higher to be installed on the same machine as the one that contains the docker engine.

4. Extract the `docker-compose-linux-ssl-<version>.tar.gz` in the desired location in the file system by using the following command:

```
sudo tar xzvf docker-compose-linux-ssl-<version>.tar.gz -C <path_to_extract>
```

   Where `<path_to_extract>` is the file system location to extract the files.

   Remember to replace the `<version>` with the actual value from the file name.

5. Edit `.env` file with a text editor, modifying the environment variable settings based on your setup. Note this file is by default hidden when using the "ls" command, to see it use "ls -a":

   - `MONGODB_DATABASE_VOLUME`: A string which represents the docker volume mapping between the file system paths from the host Linux machine and the docker container. In this case it specifies the MongoDB database file path mapping. This string contains three parts, separated by colons. The first part is the file system path to store MongoDB database files on your host machine, the second is the path in the docker container where MongoDB will save the data, and the third is the mapping option. You only need to change the first part to match your configuration.
   - `HTTPS_PORT`: The TCP/IP port number for the Signals VitroVivo Metastore Service. This will be the one targeted by your apps. Usually the 443 is used for SSL encrypted traffic.
   - `SERVER_NAME`: The domain name for the Signals VitroVivo Metastore Service Host. For instance: metastore.yourdomain.com. Make sure to use the exact name your certificate is issued to.
   - `TOKEN`: Provide the Token you want to be considered as valid on the connections to the Signals VitroVivo Metastore. This value can contain any string, try to provide a complex alphanumeric one. Look at the provided example as a reference. Remember to fill the corresponding Preference in your Spotfire® installation with the same value (review step 3.2.6).

6. Replace the sample certificate files provided with your own certificate files. You must keep the same name as the provided ones, just replace their content with the ones you have. Both certificate and private key must be included. The certificate must not require any password.

   a. Browse to nginx/certs subfolder
   b. The `cert.pem` file must contain the certicate itself. The file must begin with `-----BEGIN CERTIFICATE-----` and end with `-----END CERTIFICATE-----`
      If needed, this file can contain the full certificate chain (just concatenate them).
   c. The `key.pem` file must contain the private key related to the certificate. The file must begin with `-----BEGIN PRIVATE KEY-----` and end with `-----END PRIVATE KEY-----`

7. Under the same directory where the above file exists, execute the following shell command to load the latest metastore image into your local docker:

```
sudo docker load -i image-<version>.tar.gz
```

8. Then, within the same directory, execute the following shell command to start the Signals VitroVivo Metastore Service as a daemon process:

```
sudo docker-compose up -d
```

9. To verify that the service Docker containers have started successfully, use a web browser to access the service endpoint below to see if it returns the version number of the deployed service:

```
https://<server_name>:<https_port>/
```

Where the `<server_name>` is the name of the server which hosts the Signals VitroVivo Metastore Docker container and the one you specified in the `SERVER_NAME` within your `.env` file, the `<https_port>` represents the Signals VitroVivo Metastore Service `HTTPS_PORT` number specified in the `.env` file. If you are using the standard 443 as your HTTPS PORT there is no need to include it in the URL:

```
https://<server_name>/
```

Verify the deployment by executing the following command under Linux shell:

```
curl https://<server_name>:<https_port>/calculationProtocols && echo
```

Or by executing the following command under Windows PowerShell 6.0 or above:

```
Invoke-RestMethod https://<server_name>:<https_port>/calculationProtocols
```

Where the `<server_name>` is the name of the server which hosts the Signals VitroVivo Metastore Docker container, the `<https_port>` represents the Signals VitroVivo Metastore Service port number specified in the `.env` file.

IMPORTANT NOTE: Certificates have a valid time period, after which they expire and are no longer valid. You need to keep track of this expiry date and redeploy whenever you need to use a newer/updated certificate

3.2.5 Calculations Explorer Spotfire® Extension Deployment and Configuration

Deploy in your Spotfire® server area the corresponding Spotfire® version SDN listed under the section **Signals VitroVivo Apps Spotfire® Distribution Files (SDNs)** of Table 3-1.

Note: For additional details on deploying new packages, please refer to *Spotfire® Server and Environment Installation and Administration*.

3.2.6 Configuring Calculations Explorer Spotfire® Extensions

To share Calculations Explorer Templates across users, the Signals VitroVivo Metastore service connection details, `http://<server_name>:<port>` or `https://<server_name>:<https_port>`, must be configured in the **Signals VitroVivo Metastore > Service URL** preference for every Signals VitroVivo Spotfire® user group. If the SSL version is used, you will also need to provide the same Token used in the `.env` file within the **Signals VitroVivo Metastore > Service Security Token.**

3.3 Images Service Installation

Images Service is a standalone service application that provides an image rendering and processing service. To have a dedicated server for rendering images, install the Images Service. Otherwise skip this installation step and use the local images renderer provided by Image Discovery.

3.3.1 Prerequisites

- Windows 10 or 2008 R2 or higher

3.3.2 Installation

To install the Images Service:

1. Launch the installation executable file *(Signals Images Service-<version>.exe*) located in the Installers folder.
2. The 'Welcome to the InstallShield Wizard for Signals Images Service' will appear. Select **'Next'**.
3. From the License Agreement dialog, select "I accept the terms of the license agreement" radio button and select **'Next'**. The 'Choose Destination Location Dialog' will appear.
4. From the Destination Folder dialog, select **'Next'** to accept the default path. Alternatively, select **'Change'** to choose a different install path.
5. Select the **'Install'** button to begin the installation process.
6. Once complete, select on the **'Finish'** button.

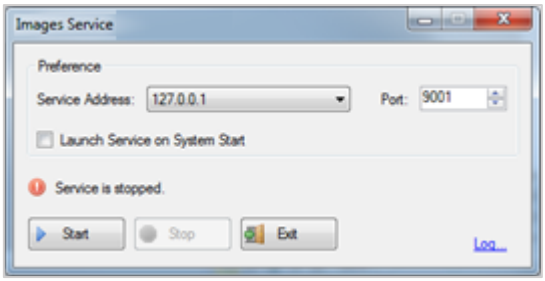
To uninstall the Images Service, select the **Uninstall Images Service** from the **Start** menu.

3.3.3 Launching the Images Service

Once the Images Service is successfully installed, manually launch it from the **Start** menu.

To launch the Images Service:

1. Define the Service Address and Port for the Image Service. The drop-down list contains all IP addresses for the current machine. Select one IP address and one port (requires one network port to provide service for Image Discovery).
2. Select the **Start** button to start the Images Service using the specified Service Address and Port.
3. Enable the **Launch Service on System Start** checkbox to launch the Images Service on startup.
4. If applicable, select the **Log** link to open the Images Service log file.



Note: If the standalone Images Service will be accessed from other machines, do not use IP address 127.0.0.1 as the service address. There may be multiple network adapters installed on other machines, especially other Servers. Each connects to a different network and has a different IP address. In this case, ensure that the Images Service uses the IP address to which the clients have access.

The local Images Service is launched by the Image Discovery extension; it requires one network port to provide service for Image Discovery.

Windows Vista and higher operating systems, by default, enable User Access Control (UAC). This may block the Images Service from accessing a port. Image Discovery will automatically launch one script to enable ports for the Images Service. If the script execution fails, open Windows command line as an administrator (right-click on command line icon, choose Run as administrator), and run the following command to enable a port for the Images Service: **netsh http add urlacl url=http://127.0.0.1:<Port Number>/ user=Everyone**

The port number should be 9251 to 9261; therefore, it is necessary to run this command ten times for port 9251 to 9261.

3.3.4 Images Renderer

By default, the Signals Images renderer is automatically set on the Image column. This preference can be modified through the Spotfire® Administration Manager.

To set AutoSetImagesRenderer preference:

1. Open the Spotfire® client and log on as a Spotfire® **Administrator**. From the **Tools** menu, select the **Administration manager** sub-menu item. The **Administration Manager** window opens.
2. Click on the **Preferences** tab.
3. Select the Group Name to which the preferences should be applied.
4. Select the **Preferences** tab in the right-hand panel. Expand the **Image Discovery** category in the tree.
5. Select the **Image** sub-category.
6. Click on the **Edit** button. The **Edit Preferences** dialog opens.
7. Set the AutoSetImagesRenderer parameter preference (True).
8. Click **OK** to save and close the dialog.
9. Click **Close** to exit the Administration Manager Signals VitroVivo Metastore Installation.

3.4 Installation of TERR™ and R Dependencies

3.4.1 Local Installation of TERR™ Packages and R Dependencies in Spotfire® Analyst

After deploying the Signals Apps but before running them, ensure all the required library dependencies are properly installed. Most of the Signals Apps dependencies are included as part of the Signals Apps distribution files, however some must be installed afterwards. TERR dependencies must be installed using the *App dependency manager*.

The installation process uses the *App dependency manager* (accessible though the settings menu, see Figure 3-1, which guides the user through the validation and installation of the required dependencies. The *App dependency manager* can assist in the installation of dependencies only when Spotfire® data functions are configured to use the locally installed TERR™ engine.

Note: The installation of packages from the dependency manager is controlled by the license ‘Signals Apps Dependency Manager’. Therefore, without a valid license the buttons to install/update are disabled, and a warning is displayed to contact the Administrator.

Figure 3-1: *How to access the App dependency manager*

Users will automatically be prompted to check their dependencies whenever there is a change to the version of Spotfire®, or when new versions of the Apps are deployed. Dismissing the message will suppress this warning until new changes are detected.

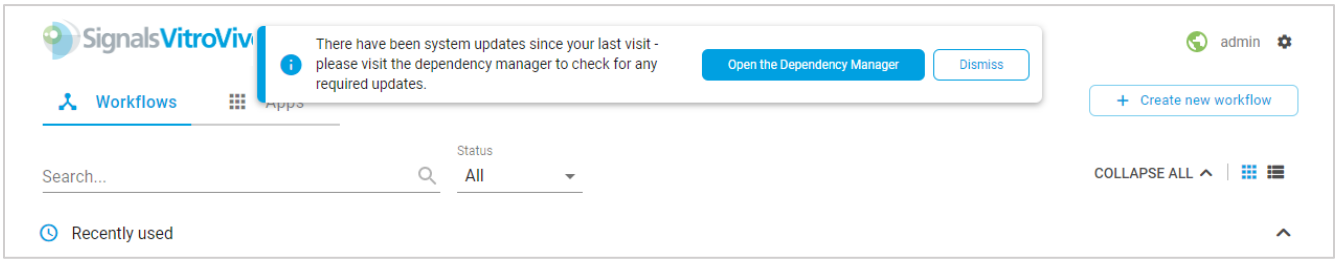


Figure 3-2: *Notification indicating the system configuration has changed and the App dependency manager must be reviewed.*

The *App dependency manager* will show tabs to display:

- Valid Apps (those whose dependencies are validated and properly installed (**Error! Reference source not found.**) and, therefore, can already be used).
- Any missing dependencies, along with additional information to ease the resolution of the detected problems. This can be one of 3 categories:
  - Spotfire® Dependencies (required TERR or Python runtimes).
  - Engines (e.g., R).
  - Packages (missing Python/R package dependencies as required by each App).

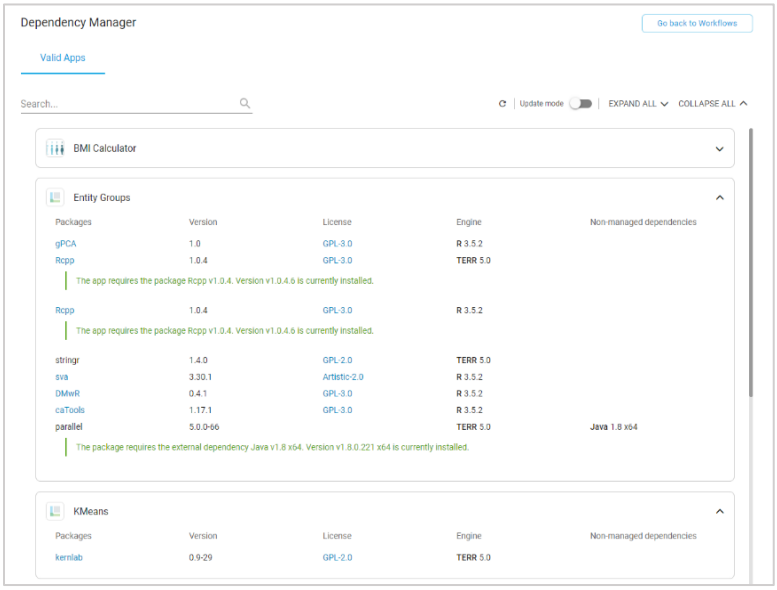


Figure 3-3: *Valid Apps status information*

The **Valid Apps** tab not only shows Apps whose dependencies are validated and properly installed, but also allows them to be updated via the “Update mode” button in the top right (Figure 3-3). When in active mode, the user can select the packages to update by selecting them in the ‘Update’ column (Figure 3-4).

Once selection is finished, the user can install the missing packages upon accepting the packages term licenses.

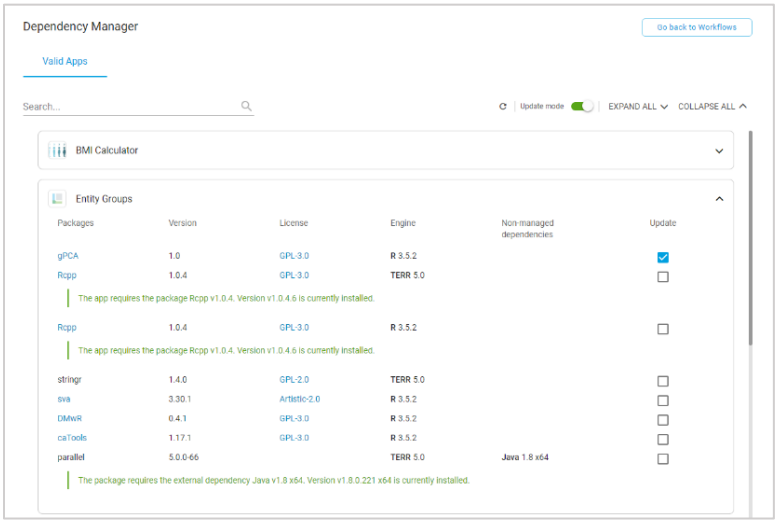


Figure 3-4: *Valid Apps update mode*

The remaining tabs show all detected issues related to an App’s installation.

- **Spotfire® Dependencies:** Some Apps require a specific engine provided by Spotfire®, such as TERR or Python. These engine dependencies will be validated by the *App Dependency Manager* but must be installed by the user if a compatible version is not found among those required. The missing engines (if the version is deemed incompatible) and the affected Signals Apps will be listed in the *App Dependency Manager, Spotfire Dependencies* tab, see *Figure 3-5*. The *Spotfire Dependencies* tab is not shown if no missing dependencies are detected.

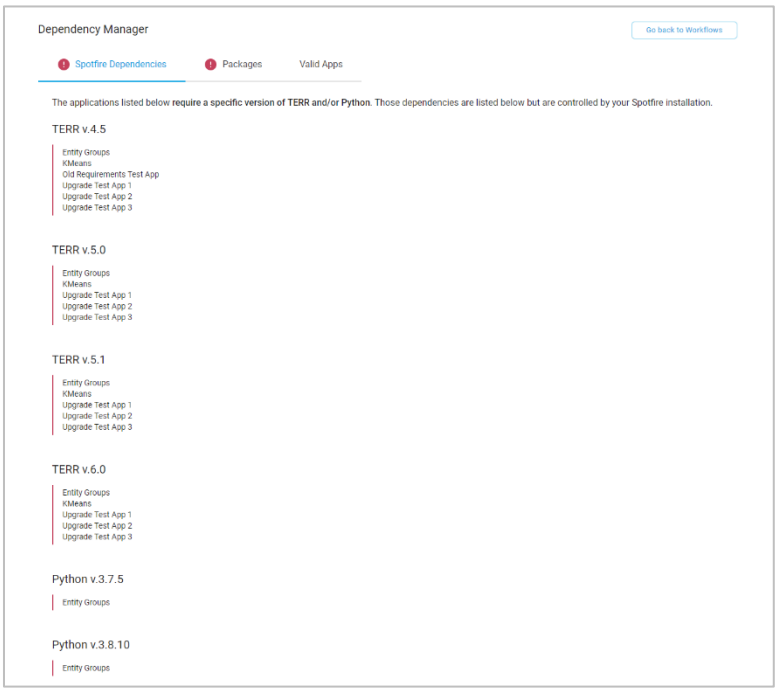


Figure 3-5: *Spotfire® Dependencies engine dependencies information*

- When a required version is not available (for instance if the engine received a hotfix which updated a dependency), the system will accept a version greater than any of the required versions (see *Figure 3-6*). For example, if an App requires TERR 4.5 or 5.0, but 5.1 is installed, then this will be deemed as compatible.
- The system will always select the set of requirements with the greatest TERR version. For example, if TERR 5.1 is installed, but an App requires TERR 4.5 or 5.0, then the system will use the set of requirements corresponding to TERR 5.0.
- Note that the same rules apply to Python, an App that requires Python 3.7.5 or Python 3.8.10 will deem Python 4.0 as compatible, and the system will use the set of requirements that correspond to Python 3.8.10.

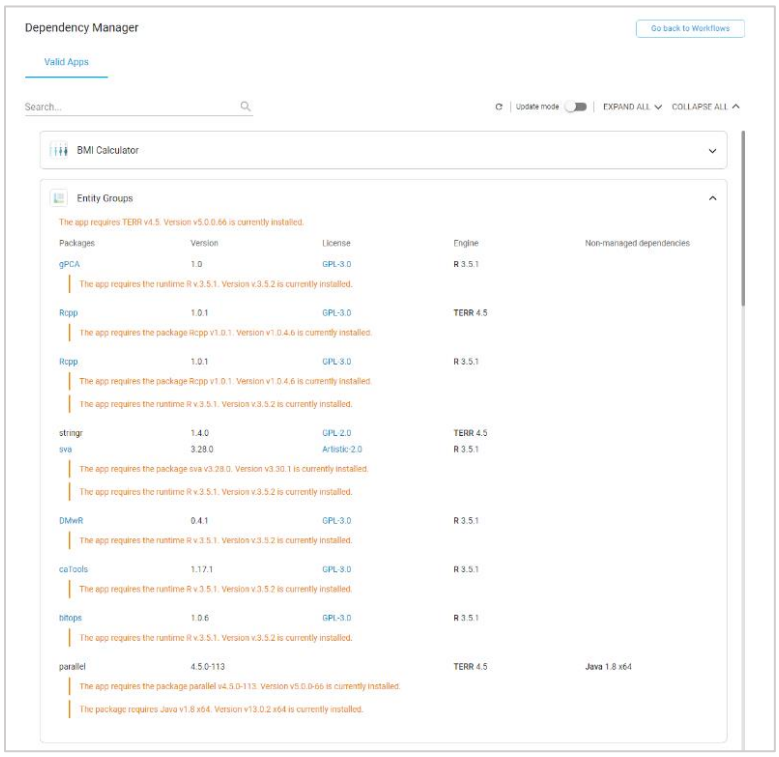


Figure 3-6: *The Entity Groups App TERR™ 4.5 requirement is deemed compatible with TERR™ 5.0 and R 3.5.1 requirement is deemed compatible with R 3.5.2*

- **Other Engines:** Some Apps might require other engines such as R. These engines will be validated by the *App dependency manager* but must be manually installed and configured by the user if there is no compatible, correctly configured version installed. The **Engines** tab is not shown if no missing TERR dependencies are detected (note that Python requirements cannot depend on an R engine).

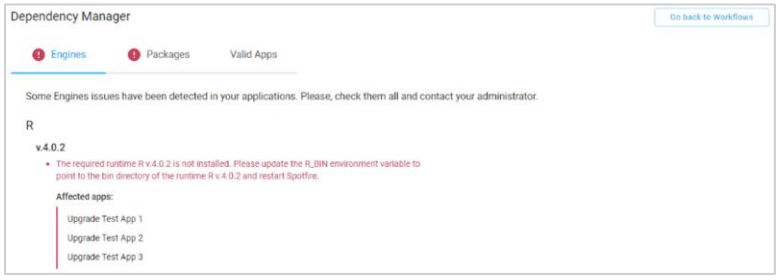


Figure 3-7: *Engines installation information*

- When the engine version required is not available, for example if R received a hotfix, the system will deem the available version as compatible (*Figure 3-6*) if the available engine version is greater than any of the required versions (i.e. if an App requires R 3.5.1 or 3.5.2, the version 3.5.3 will be deemed compatible). The system will always select the set of requirements with the greatest TERR™ version (for example, if using R 3.5.3, but an App requires R 3.5.1 or 3.5.2, then the system will use the set of requirements corresponding to R 3.5.3).

- **Packages:** Some Apps require specific R packages (installed in the local TERR™ engine or in the local R engine), or on specific Python packages (installed in the local Python engine). The required packages will be validated and automatically installed, if required, by the *App dependency manager*. The missing packages, along with the version, license and engine information will be listed for every Signals App in the **Packages** tab, *Figure 3-8*. The **Packages** tab will be hidden if no missing packages are detected. The user can install the missing packages upon accepting the packages' term licenses.

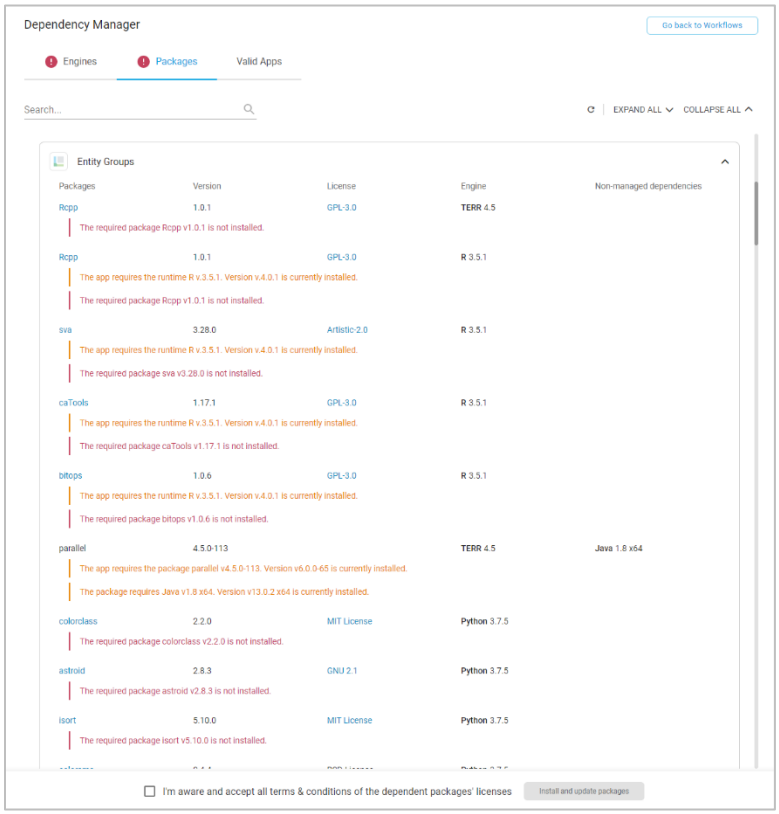


Figure 3-8: *Missing required Packages*

To install missing dependencies, follow these steps:

1. In the **Signals Apps** page, select **Settings > App dependency manager**.
2. In the *Apps dependency manager* check the Packages section to determine if there are any missing dependencies for any of the Apps.
3. If there are missing dependencies, the affected Apps will be displayed together with the list of missing packages, the required version, and the License.
4. Select the checkbox to accept any terms and conditions for these dependencies.
5. Select **Install** to download and install the packages from the CRAN repository.

App packages that have external dependencies will display them in the external dependencies column of the **Valid Apps** or the **Packages** tab (depending on their installation state, *Figure 3-3*, *Figure 3-9*, *Figure 3-10* and *Figure 3-11*). The package will also include an error/warning/informational message indicating any possible or potential issues. Note that the *Apps dependency manager* cannot solve issues related to external dependencies and require an administrator.

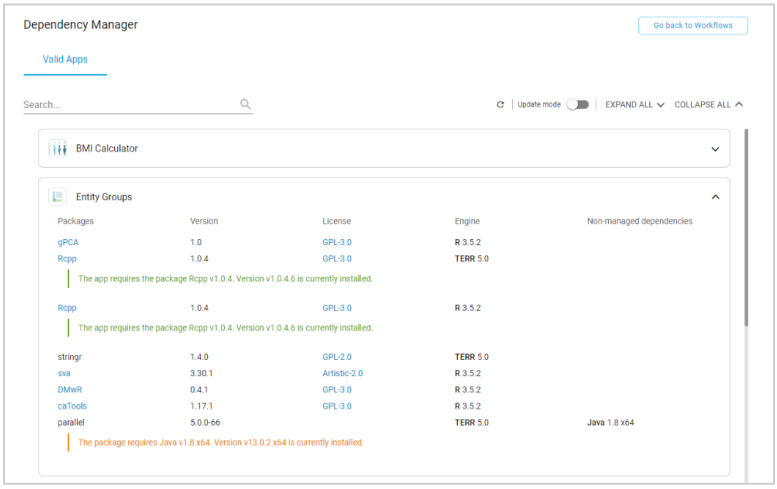


Figure 3-9: *Java external dependency with a warning message*

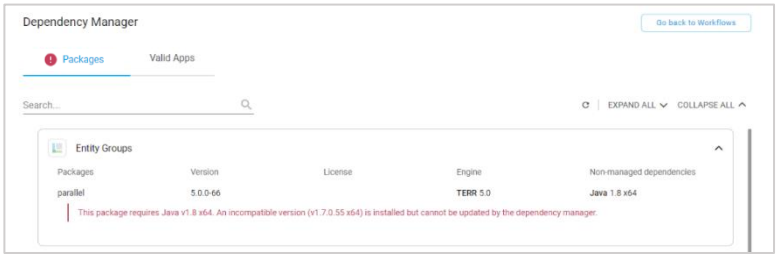


Figure 3-10: *Java external dependency with an error message*

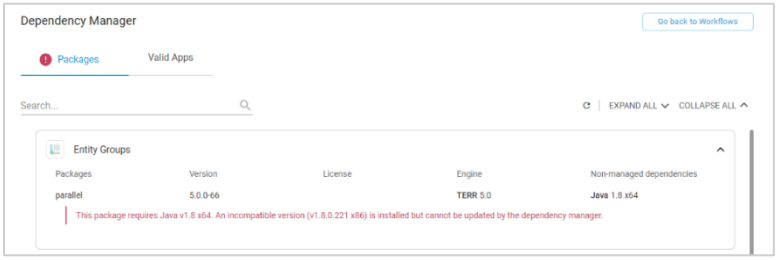


Figure 3-11: *Java external dependency with another error message*

If the Spotfire® Data Function option is configured to run data functions in Spotfire® Statistics Services, the system administrator must manually install the required dependencies in the TSSS according to the Spotfire® Installation Guide instructions. The Apps Dependency Manager will warn about this situation, see Figure 3-12.

Figure 3-12: *Apps dependency manager cannot validate dependencies when using a TSSS*

The *Apps dependency manager* supports Python modules as an App embedded resource. The installation status of Python packages will be listed for applicable Apps, allowing users to update Python packages to the latest version. Additionally, the *App dependency manager* supports the external Python installation defined via the Data functions menu, available through the **Tools > Options > DataFunctions.**

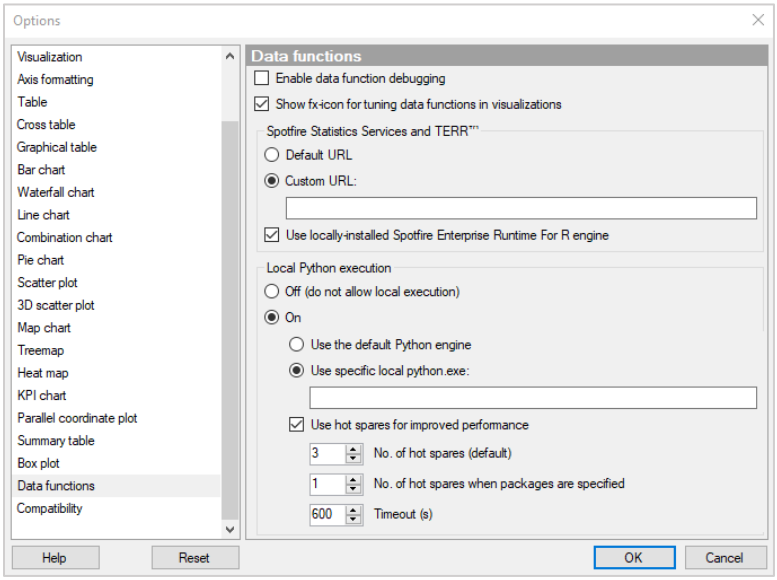


Figure 3-13: *Data functions menu for external Python installation*.

3.4.2 Installing the TERR™ Libraries on the TSSS

The main difference between Analyst and Web Player regarding TERR™ execution is that Analyst supports a local mode of execution (via a local TERR™ interpreter) and a remote mode that relies on a previously installed TSSS. Web Player only supports executing TERR™ remotely using TSSS.

Note: This installation process is not supported in a cluster environment. If using this method in a cluster environment, upload the same package on each server in the cluster. Refer to the *Package updating tools for Spotfire® Statistics Services* manual for more details.

To install the TERR™/ R dependencies in the TSSS server, complete the following:

1. Download the Signals VitroVivo vX.Y_Installers.zip archive, containing the TSSS bundle file, TERR_bundle_X.Y.Z.build.zip, from Flexera under the Signals VitroVivo section.
2. Unzip the bundle file, TERR_bundle_X.Y.Z.build.zip, in a folder in the TSSS server (e.g. in C:\Users\Administrator\Documents).
3. Configure the R_LIBS_USER system environment variable to point to the library folder of the R installation (e.g. C:\R\R-3.4.1\library). **Note:** Make sure R_LIBS_USER is defined as a system variable instead of a user specific variable.
4. Stop the TSSS service. **Note:** It is important to stop the TSSS service, if this is not done the kinetics library .dll may not be removed and the installation should be repeated after removing it manually
5. Edit the spserver.properties file adding/editing the following parameters:

   terr.restricted.execution.mode=false

   expression.service.enabled=true

   In a default installation, the full path to the spserver.properties file is:

   C:\Program Files\TIBCO\statsvcs711\<service_name>\conf
6. Run the installation script, install_for_tsss.cmd, with administrative privileges. The install_for_tsss.cmd will install the R libraries in the TERR bundle and download and install other required third-party R libraries.

Note: If installed it in a non-standard folder, modify the path to point to the TERR™ engine location of TSSS. Please note that it will take a while to download all the dependencies from CRAN. Ensure the server has internet access to CRAN.

C:\Users\Administrator\Documents\TSSS> .\install_terr_r_environments.cmd

The following messages are displayed as the libraries are installed:

Using C:\R\R-3.4.1\library for the local RinR library

Installing lib_xxx_X.X.zip from local file in TERR™

* installing *binary* package lib_xxx from "lib_xxx_X.X.zip" to " TERR_4.4.1.6/engine/library"

* checking MD5 checksums

MD5 checksums ok

Installing lib_yyy_X.X.zip from local file in TERR™

[...]

7. At the end of the script execution, a summary of the status of the installation is displayed. If an error is raised, check the message output from the installation script for more details.

   [...]

   INFO: Package lib_xxx is correctly installed in TERR™

   INFO: Package lib_yyy is correctly installed in TERR™

   […]

8. [Optional] Run the validation alone by passing the “check” argument to the script:

   C:\Users\Administrator\Documents\TSSS> .\install_terr_r_environments.cmd check

9. [Optional] Clean a previous installation by passing the “clean” argument to the script:

   C:\Users\Administrator\Documents\TSSS> .\install_terr_r_environments.cmd clean

10. After the script finishes successfully, start the TSSS service.
11. Validate that the packages are installed in TERR™. For this check the install_terr_r_environment.cmd console output.

3.4.3 Installing the itghcs Python and itghcs Resources Libraries on the TSSS

To function correctly, the **High Content Profiler** App requires the deployment of Python data functions to execute parts of the analysis.

These libraries are included in the following zip files:

- itghcs.python.resources_X.Y.Z.build.zip
- itghcs.resources_X.Y.Z.build.zip

To deploy, unzip the folders and place them within the ~ \SplusServer\data\common directory in the TSSS directory structure.

Note: Support for TSSS will be removed in future versions and replaced by TERR nodes.

3.5 Installation of Signals Inventa

To install Signals Inventa, refer to the *Revvity Signals Inventa Installation Guide* with the following modifications that are required for the use of the VitroVivo pipeline export and execution of the Out-of-the-Box (OotB) VitroVivo **4P Dose-Response IC50 Assay Workflow** on the cloud.

1. In the **Installing Signals Data Factory Services** section, the following changes from the standard installation procedure should be completed:
    - `enableSciStream`: Set this flag to `true` to use the SciStream features
    - `showPipelines`: Set this flag to `true` to show pipelines under the projects
2. In the **Import Base Information Design** subsection of the *Signals Inventa Services Installation*, append the `-b` parameter to the `create-info-design` command of siautils. This option will generate the Dose-Response Out-of-the-Box (OotB) measurement types and map definitions for the VitroVivo 4P Dose-Response IC50 Assay Workflow:

```
$ ./siautils create-info-design --url http://<ingress_host> -u super -b
```

3. Ensure that the parent-detail relationship between the Dose-Response Out-of-the-Box (OotB) measurement types is established properly. Follow the steps listed in the **Building up Relationships Between OotB Measurement Types** section in the *Revvity Signals Inventa Installation Guide*.

4. Updating Signals VitroVivo from an Earlier Version

4.1 Signals VitroVivo Apps Spotfire® Extension Packages Installation and Configuration

4.1.1 Deploying Spotfire® Extension Packages

To upgrade from earlier versions of Signals VitroVivo (previously called Signals Screening), the new Spotfire® SDN file must be deployed. Follow the instructions described in section 3.2.1

4.1.1.1 Upgrade from a Previous Version of Signals VitroVivo to Signals VitroVivo 3.5

To upgrade from version 3.4 to 3.5 no additional steps are required.

4.1.2 Setting up Spotfire® Extension Licenses

Ensure the licenses are configured as described in the section 3.2.2.

4.1.3 Signals VitroVivo Apps Spotfire® Extension Configuration

Review the configuration of the Spotfire® preferences described in the section 3.2.3.

4.2 Update of TERR™ and R Dependencies

To upgrade from earlier versions of Signals VitroVivo, it is necessary to install the R packages needed for the correct functioning of Signals VitroVivo. To do this, after the usual Spotfire® update, use the *App dependency manager* to identify and install the required packages. Follow the instructions for installing the TERR™ dependencies and R libraries as described in 3.4.

4.3 Signals VitroVivo Metastore Service

To upgrade from earlier versions of VitroVivo it is necessary to upgrade the VitroVivo Metastore Service to ensure correct functioning. The upgrading process will migrate previous Metastore data into the new Metastore version.

The upgrade instructions provided in this section assume the user has an installation performed with the **previous release**. If this is not the case, it is recommend upgrading from each version sequentially to avoid unexpected issues.

Important Note: Before performing any upgrade to the Signals VitroVivo Metastore please ensure the existing data is correctly backed up. To create a backup, follow the procedure in the Backups section below.

Important Note: Windows Metastore is no longer supported in 3.5

4.3.1 Upgrading the Signals VitroVivo Metastore Service on a Linux Machine

Follow these steps to upgrade the Metastore Service:

1. Ensure existing data is backed up.
2. Extract the docker-compose-<OS>-<version>.tar.gz in the desired location in the file system of the Metastore Service host.

   ```
   sudo tar xzvf docker-compose-linux-<version>.tar.gz -C <path_to_extract>
   ```

3. (Optional) Edit .env file with a text editor, modifying the METASTORE_PORT if configuring a different TCP/IP port than the default one.
4. (Optional) Edit .env file with a text editor, modifying the MONGODB_DATABASE_VOLUME if configuring a different value than the default one.
5. Under the same directory where the above file exists, execute the following shell command to load the new docker image into your local docker repositories.

   ```
   sudo docker load -i image-<version>.tar.gz
   ```

6. Under the same directory where the above file exists, execute the following shell command to stop the Metastore Service as a daemon process as admin:

   ```
   sudo docker-compose stop
   ```

7. Under the same directory where the above file exists, execute the following shell command to start the Metastore Service as a daemon process as admin:

   ```
   sudo docker-compose up -d
   ```

8. Execute the following shell command to check that the right Docker version is installed, the following string `*NAMES is "…informatics/sss-metastore:<version>"* should be displayed:

```
sudo docker container ls -a
```

9. To verify that the service Docker containers have started successfully, use a web browser to access the service endpoint below to see if it can return the version number of the deployed service:

```
http://<server_name>:<port>/
```

Where the `<server_name>` is the name of the server which hosts the Signals VitroVivo Metastore Docker container, the `<port>` represents the Signals VitroVivo Metastore Service port number specified in the `.env` file.

Verify the deployment by executing the following command under Linux shell:

```
curl http://<server_name>:<port>/calculationProtocols && echo
```

Or by executing the following command under Windows Powershell 6.0 or above:

```
Invoke-RestMethod http://<server_name>:<port>/calculationProtocols
```

Where the `<server_name>` is the name of the server which hosts the Signals VitroVivo Metastore Docker container, the `<port>` represents the Signals VitroVivo Metastore Service port number that you specified in the `.env` file.

5. Troubleshooting

This section contains solutions to common problems encountered during the installation of Signals VitroVivo.

5.1 Signals VitroVivo Metastore

Below are the solutions to scenarios that may be encountered when installing the Signals VitroVivo Metastore.

5.1.1 The Docker Signals VitroVivo Metastore Process Does Not Start

This could have several causes, but two common ones can be the Docker engine is not installed correctly or the user attempting to run it does not have sufficient privileges to do so.

Ensure the following:

1. Make sure Docker is installed correctly by running the command:

```
sudo docker run hello-world
```

If this fails, ensure all installation steps have been performed correctly for the operating system according to https://docs.docker.com/install/#server.

2. In case the user that must run Docker does not have admin rights follow the corresponding steps at https://docs.docker.com/install/linux/linux-postinstall/. After this run Docker from the specified user without needing root privileges:

```
docker run hello-world
```

3. In some Linux flavors manually install docker-compose. This may be done using the following commands:

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.22.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo mv /usr/local/bin/docker-compose /usr/bin/docker-compose

sudo chmod +x /usr/bin/docker-compose
```

4. In some Ubuntu 20.04 LTS the commands required to install Docker differ slightly from those on the Docker website. If the instructions on the site do not work, follow the instructions below:

```
# Install docker

sudo apt update

sudo apt install apt-transport-https ca-certificates curl software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -


# Check fingerprint

sudo apt-key fingerprint 0EBFCD88


sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"

sudo apt update


# Confirm we are downloading from the correct repo

apt-cache policy docker-ce


sudo apt install docker-ce docker-ce-cli containerd.io


# Confirm docker is up and running

sudo systemctl status docker

sudo docker run hello-world
```

5.2 Error when Pasting Data in the Editable Data Grid using the Web Player

When working with the **Editable Data Grid** App and pasting data into the table, if the length of the pasted data is too long (approximately 256KB), Spotfire® Web Player may crash and display an Ajax error as described here: https://support.revvitysignals.com/hc/en-us/articles/4408236638100-Why-am-I-seeing-an-Ajax-error-in-the-Editable-Data-Grid-.

To solve this, a setting on the Spotfire® server can be adjusted to increase the limit of the maximum size that the Web Player accepts (default 256KB). Visit the **Revvity Signals Knowledgebase** for detailed instructions here: https://support.revvitysignals.com/hc/en-us/articles/4408236509588-Information-link-with-multiple-prompts-and-

large-data-errors-out-with-The-task-could-not-be-completed-Communication-error-in-undefined-on-the-TIBCO-Spotfire-Web-Player-or-Automation-Services.

5.3 Java Installation

Below are the solutions to some scenarios that may be encountered when Java is not properly installed or configured.

5.3.1 Curve Fitting Does Not Work from Within the CE Templates

This could have several causes, but the most common would be the JRE prerequisite is missing or incorrectly configured. This is required for using the parallel library in TERR™ and when missing the curve fitting execution will not work correctly in trellised plots.

Ensure the following:

1. The 64-bit version for JRE version 8.0 or higher is installed.
2. JAVA_HOME environment variable is available and pointing to the jre (this can usually be configured in **System Properties > Advanced > Environment Variables > Edit System Variable**).
3. Follow the indications to set JAVA_HOME from TERR™ available at: https://docs.tibco.com/pub/enterprise-runtime-for-R/latest/doc/html/TIB_terr_Documentation/doc/topics/setting_java_home.html.

6. Backups

To mitigate potential damage due to hardware or software failure, it is highly recommended that a backup for the information that has been created in the system and could cause significant damage if lost, is performed. Within Signals VitroVivo the following elements should be backed up:

- App Workflows
- File import templates
- Metastore information

6.1 Backing Up and Restoring the App Workflows

The App Workflows information is stored within the Spotfire® library system within the directory configured in the previously described section, Configuring Signals VitroVivo App Workflows Storage.

To create a backup:

1. Open Spotfire® Analyst as an **administrator**.
2. Open **Tools > Library administration.**
3. Navigate to and select the Workflow folder defined in the Configuring the Signals VitroVivo Input Data Format Storage section.
4. Select “Export” and select a unique export name. Spotfire® will create a zip file with the contents of the folder that will be stored in the server.

To restore an existing backup:

1. Open Spotfire® Analyst as an **administrator**.
2. Open **Tools > Library administration.**
3. Navigate to and select the Workflow folder defined in the Configuring the Signals VitroVivo Input Data Format Storage section.
4. Select “Import” and select an existing Workflows folder backup to restore.

Note: Changes will not be available until users log in again.

For more information on restoring a deleted Spotfire® Library please check the available information on the support website.

6.2 Backing Up and Restoring the Shared File Import Templates

The shared file import templates are stored within the Spotfire® library system within the directory configured in the previously described section, Configuring the Signals VitroVivo Input Data Format Storage.

To create a backup of file import templates Workflows:

1. Open Spotfire® Analyst as an **administrator.**
2. Open **Tools > Library administration.**
3. Navigate to and select the Workflow folder defined in the Configuring the Signals VitroVivo Input Data Format Storage section.
4. Click on “Export” and select a unique export name. Spotfire® will create a zip file with the contents of the folder that it will store in the server.

To restore an existing backup:

1. Open Spotfire® Analyst as an **administrator.**
2. Open **Tools > Library administration.**
3. Navigate to and select the Workflow folder defined in the Configuring the Signals VitroVivo Input Data Format Storage section.

For more information on restoring a deleted Spotfire® Library please check the available information on the support website.

6.3 Backing Up and Restoring the Metastore Information

The Metastore information is stored within the MongoDB database in the Metastore Docker instance.

6.3.1 Backing Up Metastore Information

To back up the Metastore information first create a directory in the file system to store the backup. For this example, use: <Filesystem path>/mongo_backup/. After this, from the same directory where the Metastore file exists, execute the following commands:

1. docker exec -it mongodb bash
2. mkdir mongo_backup
3. mongodump --out /mongo_backup
4. exit

5. docker cp mongodb:/mongo_backup/ <Filesystem path>/mongo_backup/

6.3.2 Restoring the Metastore Information

To restore the Metastore information from the same directory where the Metastore file exists, execute the following commands:

1. docker cp <Filesystem path>/mongo_backup/ mongodb:/
2. docker exec -it mongodb bash
3. cd mongo_backup/
4. mongorestore signals_metastore/calculationprotocols.bson
   mongorestore signals_metastore/platedesign.bson
   mongorestore signals_metastore/calculations.bson

7. Appendices

7.1 Appendix Web Player and TSSS Deploy Addendum

Deployment of Signals VitroVivo™ on the Web Player requires the following components:

- Spotfire® Server with Web Player
- TSSS which is required for the execution of data functions when running on the Web Player

7.1.1 Web Player System Requirements

Category	Requirement
Hardware	
Processor	• Minimum: 2 Cores, 2 GHz Recommended: 4 Cores or more (Intel Core i5 or equivalent), 2+ GHz, 64-bit
RAM	• Minimum: 8 GB • Recommended: 16 GB or greater **Note:** Large datasets can require more RAM.
Hard Disk Space	10 GB is recommended for installation and normal use.
Display	• Minimum: 1024x768 pixel resolution, 16 or 32-bit color depth. • Recommended: 1366x768 pixel resolution or higher, 16 or 32-bit color depth. **Note:** For functionality that require hardware acceleration (such as the 3D plot), DirectX 9 or higher and a compatible graphics card is required.
Software	
Operating System	Windows 10 64-bit

Spotfire® Software	• Spotfire® Web Player 12.0.x server, see https://docs.tibco.com/pub/spotfire/general/sr/sr/topics/spotfire_web_player_12_0.html for Spotfire® Web Player system requirements Spotfire® Web Player 14.0.x server, see https://docs.tibco.com/pub/spotfire/general/sr/sr/topics/spotfire_web_player_14_0.html for Spotfire® Web Player system requirements

7.1.2 TSSS System Requirements

Category	Requirement
Hardware	
Processor	• Minimum: 2 Cores, 2 GHz • Recommended: 4 Cores or more (Intel Core i5 or equivalent), 2+ GHz, 64-bit
RAM	• Minimum: 8 GB • Recommended: 16 GB or greater **Note:** Large datasets can require more RAM.
Hard Disk Space	• 10 GB is recommended for installation and normal use.
Display	• Minimum: 1024x768 pixel resolution, 16 or 32-bit color depth. • Recommended: 1366x768 pixel resolution or higher, 16 or 32-bit color depth. For functionality that requires hardware acceleration (such as the 3D plot), DirectX 9 or higher a compatible graphics card is required.
Software	
Operating System	• Windows 10 64-bit
Spotfire® Statistical Services	• Spotfire® Statistical Services 12.0.4 server, see https://docs.tibco.com/pub/spotfire/general/sr/sr/topics/spotfire_statistics_services_12_0.html for Spotfire® Statistical Services system requirements
TERR™ Version	• TERR 6.0.3, see https://docs.tibco.com/pub/enterprise-runtime-for-R/6.0.0/doc/html/TIB_terr_Documentation/doc/topics/tibco_enterprise_runtime_for_r_system_requirements.html for Spotfire® Server system requirements