



**PerkinElmer Signals VitroVivo™ 3.2.0**

---

**Installation Guide**

---

Powered by TIBCO Spotfire®

**Last Updated: September 23, 2022**



## Table of Contents

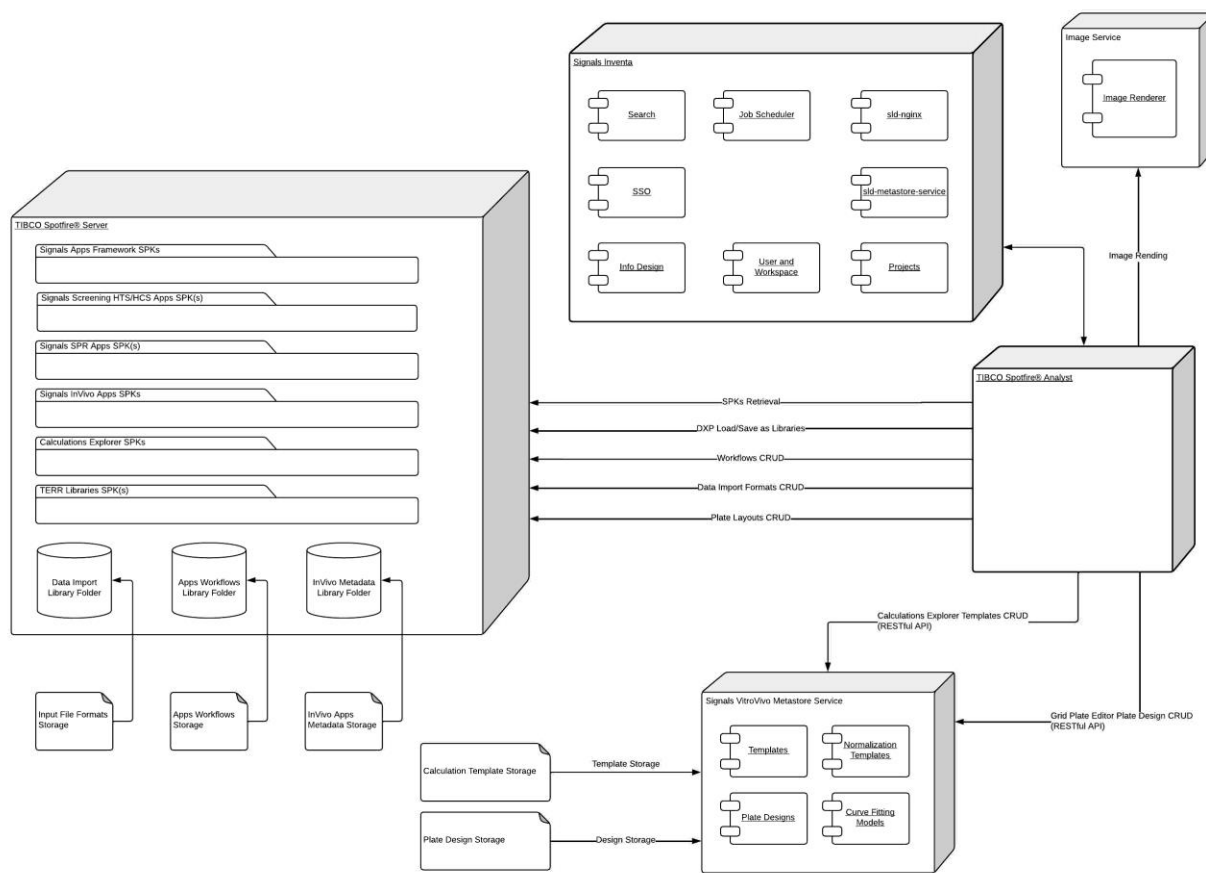
|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b>  | <b>4</b>  |
| <b>2</b> | <b>Prerequisites</b>   | <b>5</b>  |
| <b>3</b> | <b>Installation</b>  | <b>5</b>  |
| 3.1      | Installation Overview  | 5         |
| 3.2      | Signals VitroVivo Apps TIBCO Spotfire® Extension Packages Installation and Configuration | 7         |
| 3.2.1    | Deploying TIBCO Spotfire® Extension Packages   | 7         |
| 3.2.2    | Setting up TIBCO Spotfire® Extension Licenses  | 9         |
| 3.2.3    | Signals VitroVivo Apps TIBCO Spotfire® Extension Configuration                           | 10        |
| 3.2.4    | Signals VitroVivo Metastore Service  | 13        |
| 3.2.5    | Calculations Explorer Spotfire® Extension Deployment and Configuration                   | 15        |
| 3.2.6    | Configuring Calculations Explorer Spotfire® Extensions                                   | 16        |
| 3.3      | Images Service Installation  | 16        |
| 3.3.1    | Prerequisites  | 16        |
| 3.3.2    | Installation   | 16        |
| 3.3.3    | Launching the Images Service   | 16        |
| 3.3.4    | Images Renderer  | 17        |
| 3.4      | Installation of TERR™ and R Dependencies   | 18        |
| 3.4.1    | Local Installation of TERR™ Packages and R Dependencies in TIBCO Spotfire® Analyst       | 18        |
| 3.4.2    | Installing the TERR™ Libraries on the TSSS   | 24        |
| 3.4.3    | Installing the itghcs Python and itghcs Resources Libraries on the TSSS                  | 26        |
| 3.5      | Installation of TERR™ Packages   | 26        |
| 3.5.1    | Installing Signals VitroVivo TERR™ Libraries (No Dependencies)                           | 27        |
| 3.5.2    | Installing Signals VitroVivo Client TERR™ Libraries and Dependencies                     | 27        |
| 3.6      | Installation of Signals Inventa  | 29        |
| <b>4</b> | <b>Updating Signals VitroVivo from an Earlier Version</b>                                | <b>30</b> |
| 4.1      | Signals VitroVivo Apps Spotfire® Extension Packages Installation and Configuration       | 30        |
| 4.1.1    | Deploying TIBCO Spotfire® Extension Packages   | 30        |
| 4.1.2    | Setting up TIBCO Spotfire® Extension Licenses  | 30        |
| 4.1.3    | Signals VitroVivo Apps TIBCO Spotfire® Extension Configuration                           | 30        |
| 4.2      | Update of TERR™ and R Dependencies   | 30        |
| 4.3      | Signals VitroVivo Metastore Service  | 31        |
| 4.3.1    | Upgrading the Signals VitroVivo Metastore Service on a Linux Machine                     | 31        |
| 4.3.2    | Upgrading the Signals VitroVivo Metastore Service on a Windows Machine                   | 32        |
| <b>5</b> | <b>Troubleshooting</b>   | <b>33</b> |
| 5.1      | Signals VitroVivo Metastore  | 33        |
| 5.1.1    | The Docker Signals VitroVivo Metastore Process Does Not Start                            | 33        |
| 5.2      | Error when Pasting Data in the Editable Data Grid using the Web Player                   | 34        |
| 5.3      | Java Installation  | 34        |
| 5.3.1    | Curve Fitting Does Not Work from Within the CE Templates                                 | 35        |
| <b>6</b> | <b>Backups</b>   | <b>35</b> |
| 6.1      | Backing Up and Restoring the App Workflows   | 35        |
| 6.2      | Backing Up and Restoring the Shared File Import Templates                                | 36        |
| 6.3      | Backing Up and Restoring the Metastore Information                                       | 36        |
| 6.3.1    | Backing Up Metastore Information   | 36        |
| 6.3.2    | Restoring the Metastore Information  | 36        |
| <b>7</b> | <b>Appendices</b>  | <b>37</b> |
| 7.1      | Appendix In Vivo Data Model Designer App   | 37        |
| 7.1.1    | Concepts   | 39        |
| 7.1.2    | Custom Attributes  | 39        |
| 7.1.3    | Compounds  | 41        |
| 7.1.4    | Formulations   | 41        |
| 7.1.5    | Scheduling & Events  | 42        |

|       |   |    |
|-------|---|----|
| 7.2   | Appendix In Vivo Integration with Signals Notebook.....       | 42 |
| 7.2.1 | Creating Admin Define Tables (ADTs) in Signals Notebook ..... | 43 |
| 7.2.2 | Adding an ADT to a Signals Notebook Experiment .....          | 46 |
| 7.2.3 | Adding a Compound Table to a Signals Notebook Experiment..... | 47 |
| 7.3   | Appendix Web Player and TSSS Deploy Addendum.....             | 48 |
| 7.3.1 | Web Player System Requirements .....                          | 48 |
| 7.3.2 | TSSS System Requirements .....                                | 49 |

## 1 Introduction

Signals VitroVivo™ contains the following components:

- TIBCO 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
  - 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 *PerkinElmer Signals VitroVivo System Requirements* document for further details.

1. **TIBCO Spotfire® Server** and **Analyst Client** have both been installed.
  - a. This process is explained in detail in the *TIBCO Spotfire® Installation and Configuration Manuals* and the *TIBCO Spotfire® Deployment and Administration Manual*.
  - b. You must be a TIBCO 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 *PerkinElmer 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 *PerkinElmer 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 TIBCO 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 must be deployed on the TIBCO 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. License setup of the Signals VitroVivo Apps Spotfire® extensions [\*]
3. Configuration of Signals VitroVivo input file formats and Apps Workflow storage [\*]
4. Installation of Images Service [\*]
5. Deployment of Signals VitroVivo Metastore Service (requires Linux server sudo access if deployed in Linux)
6. Deployment and configuration of the Calculations Explorer Spotfire® extension [\*]
7. Installation and configuration of Signals Inventa (refer to *PerkinElmer Signals Inventa Installation Guide*)

[\*] Requires TIBCO Spotfire® administration privileges

The Signals VitroVivo release goods package includes the following items:

| File Name  | Description  |
|--|--|
| <b>Signals VitroVivo Apps Spotfire® Distribution Files (SDNs)</b>                |  |
| PerkinElmer-VitroVivo.sdn  | The Spotfire® SDN of Signals Apps  |
| <b>Files for Images Service</b>  |  |
| Image Discovery-Service-<version>.exe  | Images Service installer   |
| <b>Files for Signals VitroVivo Metastore Service</b>                             |  |
| docker-compose-linux-<version>.tar.gz<br>docker-compose-windows-<version>.tar.gz | The tar.gz file which contains the files necessary to bootstrap the Signals VitroVivo Metastore includes: <ul style="list-style-type: none"> <li>• .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</li> <li>• docker-compose.yml: The YAML configuration file for the docker-compose command line to start the Docker containers for Signals VitroVivo Metastore Services</li> </ul> |

**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 *PerkinElmer Signals Inventa Installation Guide*.

## 3.2 Signals VitroVivo Apps TIBCO Spotfire® Extension Packages Installation and Configuration

### 3.2.1 Deploying TIBCO 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 TIBCO Spotfire® Server to enable the *PerkinElmer Signals Apps Tool* which opens the *Signals Apps* catalog.

Deploy in the TIBCO 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:

- Image Import <version>
- Image Discovery <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>
- PerkinElmer Calculations Explorer <version>
- PerkinElmer Signals Analytics Apps <version>
- PerkinElmer Signals Analytics Apps for Screening <version>
- PerkinElmer Signals Analytics Common <version>
- PerkinElmer Signals Client for .NET <version>
- PerkinElmer Signals Groups <version>
- PerkinElmer SPR Alignment App <version>
- PerkinElmer SPR Blank Subtraction App <version>
- PerkinElmer SPR Common <version>
- PerkinElmer SPR Cropping App <version>
- PerkinElmer SPR Data Import App <version>
- PerkinElmer SPR Data Import SDK App <version>
- PerkinElmer SPR Data Readers <version>
- PerkinElmer SPR Export Report App <version>
- PerkinElmer SPR Hit Selection App <version>

- PerkinElmer SPR Kinetic Analysis App <version>
- PerkinElmer SPR MultiCycle Kinetic Analysis App <version>
- PerkinElmer SPR PLA App <version>
- PerkinElmer SPR QC-QA App <version>
- PerkinElmer SPR RAC App <version>
- PerkinElmer SPR Reference App <version>
- PerkinElmer SPR Referencing App <version>
- PerkinElmer SPR Relative Potency App <version>
- PerkinElmer SPR Report Points App <version>
- PerkinElmer SPR Single Cycle Kinetics App <version>
- PerkinElmer SPR Solvent Correction App <version>
- PerkinElmer SPR Steady State Analysis App <version>
- PerkinElmer SPR TraceDrawer Export App <version>
- PerkinElmer SPR Zeroing App <version>
- PerkinElmer.Signals.InVivo.BaselineCapture <version>
- PerkinElmer.Signals.InVivo.BusinessRules <version>
- PerkinElmer.Signals.InVivo.Common <version>
- PerkinElmer.Signals.InVivo.DataModelDesigner <version>
- PerkinElmer.Signals.InVivo.DosePreparation <version>
- PerkinElmer.Signals.InVivo.DoseVerification <version>
- PerkinElmer.Signals.InVivo.MassSpec <version>
- PerkinElmer.Signals.InVivo.PKParameters <version>
- PerkinElmer.Signals.InVivo.SequenceOfEvents <version>
- PerkinElmer.Signals.InVivo.StudyDesigner <version>
- PerkinElmer Signals VitroVivo Product <version>
- SciStream <version>
- Signals Notebook Spotfire Opener <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 *TIBCO Spotfire® Server and Environment Installation and Administration*.



### 3.2.2 Setting up TIBCO 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 TIBCO Spotfire's® instructions on how to edit group license rights. Both the Apps and features licenses are categorized as TIBCO 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
- DataModelDesignerLicense: For those groups that will create Study Templates

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 TIBCO 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 TIBCO 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 TIBCO 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 TIBCO 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 TIBCO 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 TIBCO 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 TIBCO 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 TIBCO 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 TIBCO 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 TIBCO 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 optionally 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.

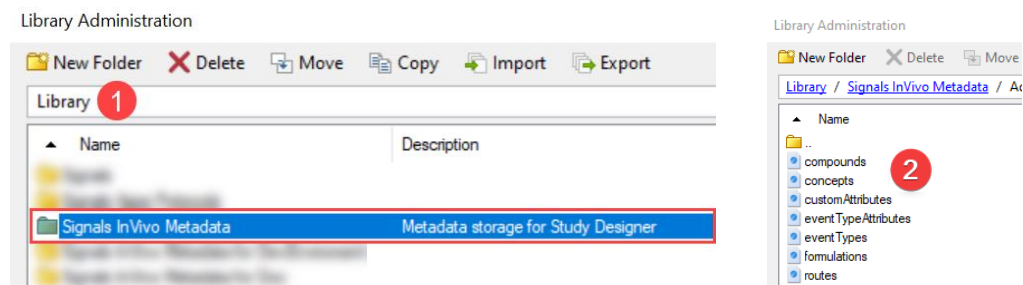
### 3.2.3.7 Configuring Library and Spotfire® Preferences for the In Vivo Domain

Before executing an **In Vivo Workflow**, at least one **Study Template** with controlled vocabulary should be defined and saved as In Vivo metadata using the **Data Model Designer App (DMD App)**. Note that multiple Study Templates may be created. Users will need to select the Study Template to be applied in the **Study Designer App** step of a Workflow.

Create a new folder in the Spotfire® library to store the In Vivo metadata, as shown in **1** and described below.

1. Log in as a Spotfire® **Administrator**.
2. Navigate to **Tools > Library administration**.
3. Create a folder (any name may be specified, i.e. 'Signals InVivo Metadata').
4. Ensure that your users/user-groups can access the folder with read-write permissions enabled.

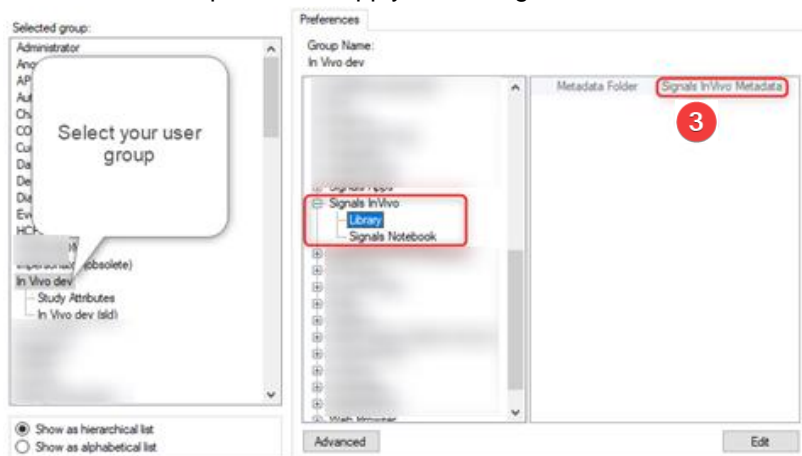
This folder is populated by the DMD App to store the In Vivo metadata with different subfolders, one per Study Template. Note the example folder contents of a defined Study Template in **2**.



Edit the 'Signals InVivo\Library' preference and specify the library folder name created in the **Library Configuration** section, as shown in **3** and described below.

1. Log in as a Spotfire® **Administrator**.
2. Navigate to **Tools > Administration Manager > Preferences**.
3. Select a user group from the left-hand panel.
4. Select **Signals InVivo > Library** in the right-hand panel.
5. Select **Edit**.

6. Edit the **Metadata Folder** name and select **OK**.
7. Restart Spotfire® to apply the changes.



### 3.2.4 Signals VitroVivo Metastore Service

#### 3.2.4.1 Deploying the Signals VitroVivo Metastore Service on a Linux Machine

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 installed the docker engine.

Follow the steps below to deploy the Signals VitroVivo Metastore Service with docker-compose:

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 substitute 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 scenario. 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 within the docker container. In this case it specifies the MongoDB database file path mapping. This string contains three parts, each being separated by

the colon. The first part is the file system path to store MongoDB database files on your host machine, the second part is the path in the docker container where MongoDB will save the data, and the third part is the mapping option. You only need to change the first part to meet your needs.

- `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 start the Signals VitroVivo Metastore Service as a daemon process:

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

7. 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 Windows Machine

1. Log on to the Windows Server machine.
2. Install Docker Desktop at <https://docs.docker.com/docker-for-windows/install/>.
  - a. Run Docker Desktop.
  - b. In some Windows versions Docker selects Windows containers by default, switching to Linux containers is required.
  - c. In some cases, an error regarding Hyper-V will be thrown, to fix follow these steps:
    - i. Open “**Windows security**”
    - ii. Open “**App & Browser control**”
    - iii. Select “**Exploit protection settings**” at the bottom
    - iv. Switch to “**Program settings**” tab
    - v. Locate “**C:\WINDOWS\System32\vmcompute.exe**” in the list and expand it
      - If it is not present in the list, add it manually.
    - vi. Select “**Edit**”

- vii. Scroll down to “**Code flow guard (CFG)**” and uncheck “**Override system settings**”
    - If the checkbox is not checked, follow the procedure in “d”.
  - viii. Start vmcompute from powershell “*net start vmcompute*”.
    - If vmcompute fails to start, follow the procedure in “d”.
  - d. If after performing the steps in “c” Hyper-V is still not working, proceed with the following steps:
    - ix. Disable Hyper-V and restart the computer.
    - x. Enable Hyper-V and restart the computer again.
    - xi. Follow the procedure in “c” again.
3. Extract the `docker-compose-windows-<version>.tar.gz` with 7zip in the desired location in the file system.
  4. Edit `.env` file with a text editor, modifying the environment variable settings based on your scenario:
    - a. `METASTORE_PORT`: The TCP/IP port number for the Signals VitroVivo Metastore Service.

**Note:** On Windows Signals VitroVivo Metastore uses Docker volume.
  5. Under the same directory where the above file exists, execute the following shell command to start the Signals VitroVivo Metastore Service as a daemon process as administrator:

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

- a. To verify that the service Docker containers have started successfully, you can 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>/
```

- b. 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.
- c. Or by executing the following command under Windows PowerShell 6.0 or above:

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

- d. 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.

6. Find the IP of the host by running:

```
docker ps
```

and noting the IP corresponding to the Docker instance running the server.

### 3.2.5 Calculations Explorer Spotfire® Extension Deployment and Configuration

Deploy in your TIBCO 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 *TIBCO 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>`, must be configured in the **Signals VitroVivo Metastore > Service URL** preference for every Signals VitroVivo TIBCO Spotfire® user group.

## 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 (*PerkinElmer Images Service-<version>.exe*) located in the Installers folder.
2. The 'Welcome to the InstallShield Wizard for PerkinElmer 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.

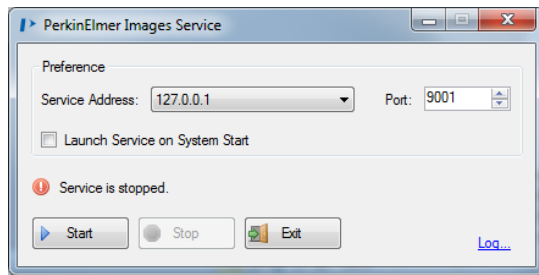
### 3.3.3 Launching the Images Service

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 PKI 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.

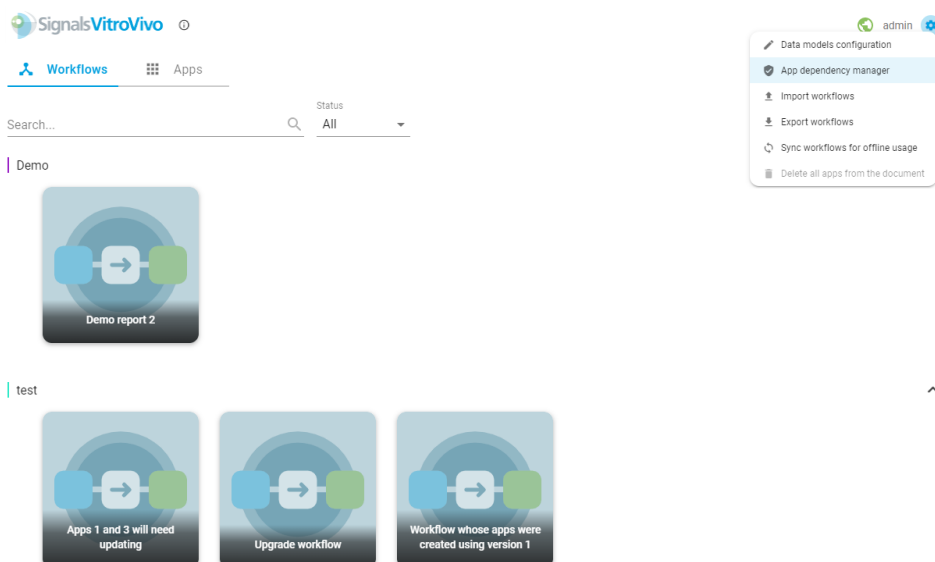
## 3.4 Installation of TERR™ and R Dependencies

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

After deploying the Signals Apps but before running them, ensure all the required dependencies are properly installed. TERR dependencies must be installed using the *App dependency manager*.

The installation process uses the *App dependency manager* (accessible through 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 TIBCO Spotfire® data functions are configured to use the locally installed TERR™ engine.

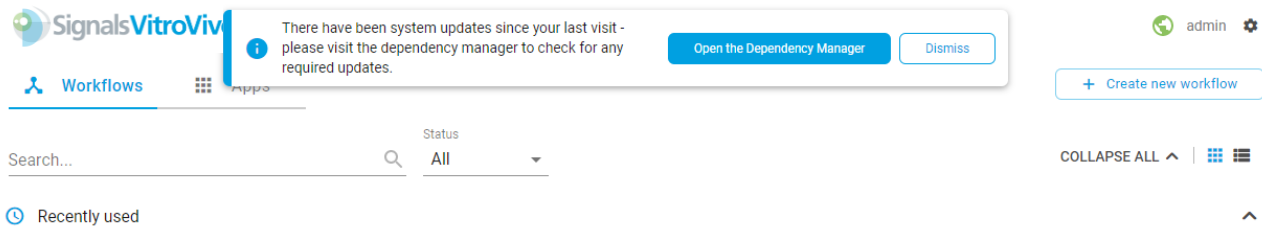
**Note:** The installation of packages from the dependency manager is controlled by the license ‘Signals Apps Dependency Manager’. Therefore, without a valid license the buttons to install/update are disabled, and a warning is displayed to contact the Administrator.



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

If the system configuration has changed (i.e. the Spotfire® version has changed, a new App with requirements has been added, or an existing App with requirements has changed versions), a notification will be shown directing the user to the *App dependency manager*. Accessing the *App dependency manager* or dismissing the notification will prevent the notification from being shown again (until new system changes are detected) and the user is responsible for taking the necessary actions in the Dependency Manager.

**Note:** Dismissing the notification will prevent the notification from showing again, however the dependencies will **not** be installed, which may cause problems when using the product.

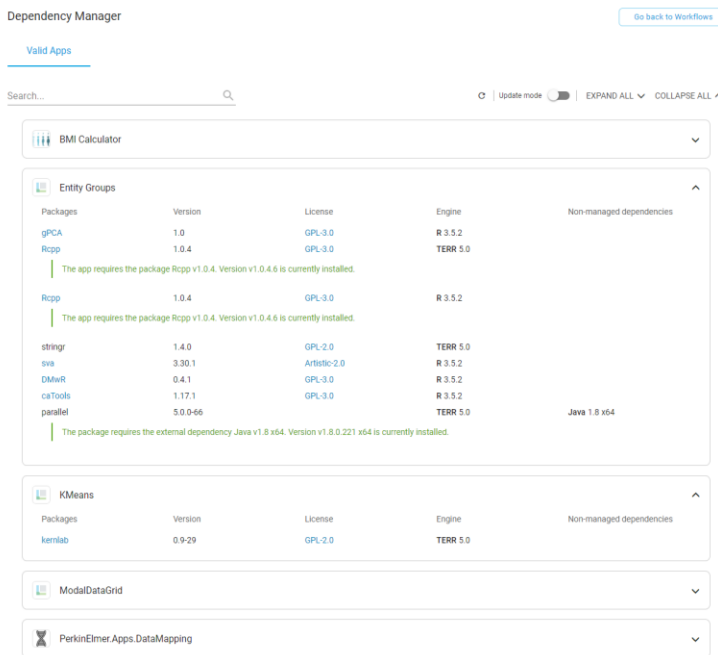


**Figure 3-2:** Notification indicating the system configuration has changed and the App dependency manager must be reviewed

If data functions are executed remotely by a TIBCO Spotfire® Statistics Services server, then the installation of dependencies must be handled by a TIBCO Spotfire® Statistics Services Administrator. Please refer to the *Signals Apps Installation User Guide* for further details.

The *App dependency manager* will list both:

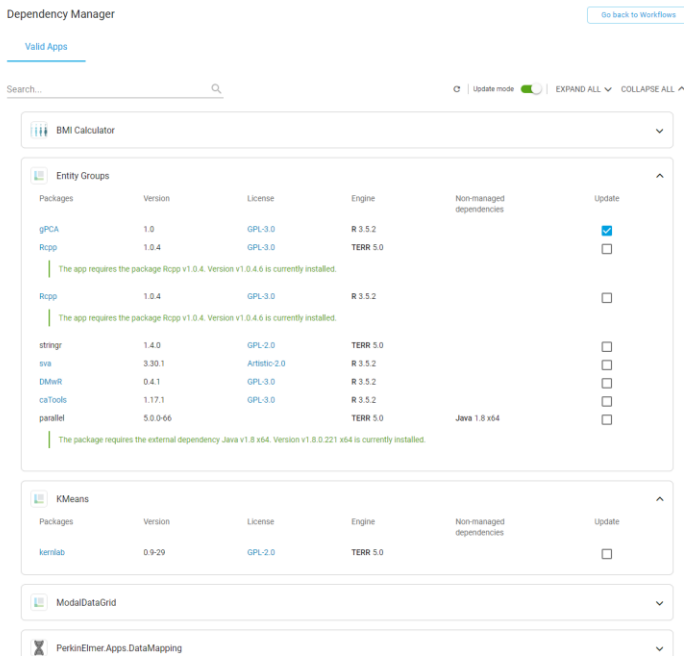
- Apps whose dependencies are validated and properly installed (Figure 3-2) and, therefore, can already be used.
- The missing dependencies along with additional information to ease the resolution of the detected problems.



**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). Selecting it will add an “Update” column that will allow the user to select the packages to update (Figure 3-4). Note that if the same package is present in several Apps, all its appearances will be de/selected automatically when selecting one.

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.

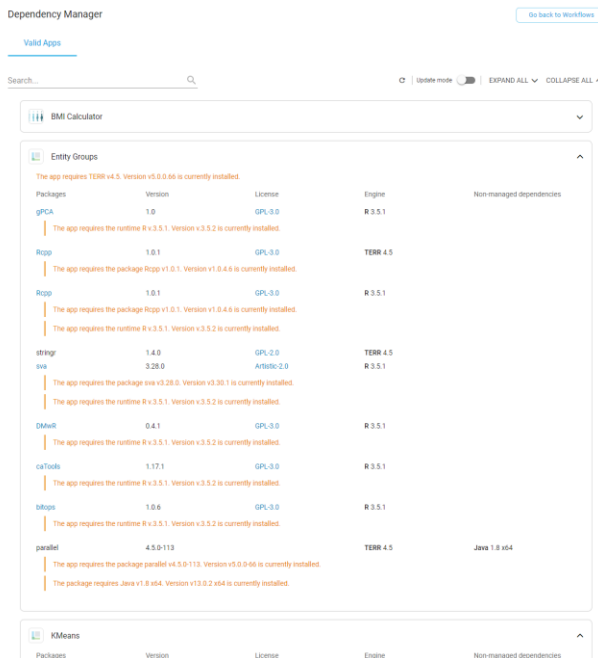
There are several different types of potential missing dependencies:

- TERR™ Engine:** Some Apps require a specific TIBCO Spotfire® TERR™ engine version. The TERR™ 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 TERR™ engines (if the TERR™ version is deemed incompatible) and the affected Signals Apps, will be listed in the *App dependency manager* **TERR** tab, (Figure 3-5). The **TERR** tab is hidden if no missing TERR™ dependencies are detected.



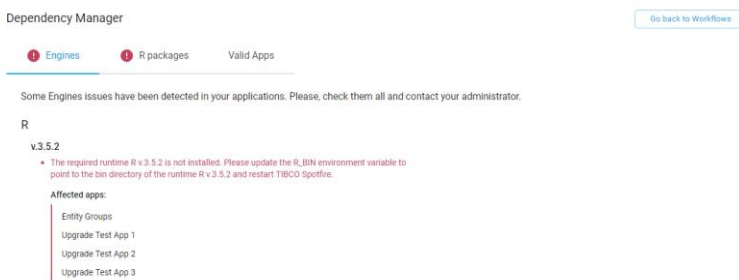
**Figure 3-5:** TERR™ engine dependencies information

- When the TERR™ version required is not available, for instance if TERR™ received a hotfix, the system will deem the available version as compatible (Figure 3-6) if the available TERR™ version is greater than any of the required versions. For example, if an App requires TERR™ 4.5, the version 5.1 will be deemed compatible. The system will always select the set of requirements with the greatest TERR™ version (for example, if using TERR™ 5.1, but an App requires TERR™ 4.5, then the system will use the set of requirements corresponding to TERR™ 5.1).



**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

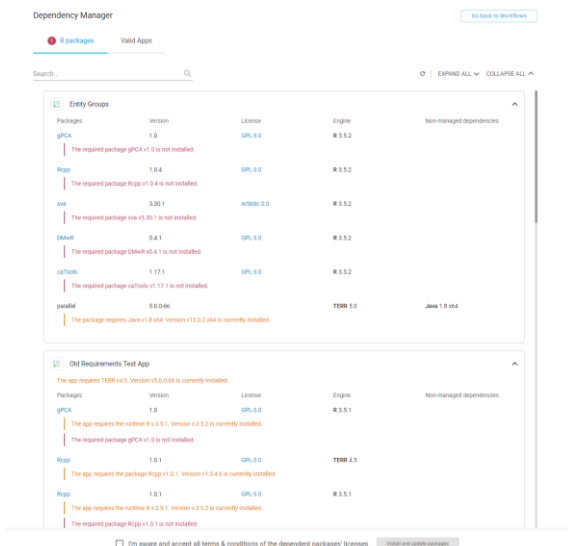
- Other Engines:** Some Apps might require other engines such as R. These engines will be validated by the App dependency manager but must be manually installed by the user if there is no compatible version installed, the required versions are not installed, or if additional configuration is required, such as the definition and setup of environment variables, see Figure 3-7. The **Engines** tab is hidden if no missing dependencies are detected.



**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).

- R packages:** Some Apps require specific R packages installed in the local TERR™ engine or in the local R engine. The required R packages will be validated and automatically installed, if required, by the *App dependency manager*. The missing R packages, along with the version, license and engine information will be listed for every Signals App in the **R packages** tab, *Figure 3-8*. The **R packages** tab will be hidden if no missing R packages are detected. The user can install the missing packages upon accepting the R packages term licenses, **Error! Reference source not found.**



**Figure 3-8: Missing required R packages**

To install missing dependencies, follow these steps:

1. In the **PerkinElmer Signals Apps** page, select **Settings > App dependency manager**.
2. In the *Apps dependency manager* check the R 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 **R 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.

Dependency Manager [Go back to Workflows](#)

[Valid Apps](#)

Search...

Update mode  EXPAND ALL COLLAPSE ALL

**BMI Calculator**

**Entity Groups**

| Packages   | Version  | License      | Engine   | Non-managed dependencies |
|--|----------|--------------|----------|--------------------------|
| gPCA   | 1.0      | GPL-3.0      | R 3.5.2  |                          |
| Rcpp   | 1.0.4    | GPL-3.0      | TERR 5.0 |                          |
| The app requires the package Rcpp v1.0.4. Version v1.0.4.6 is currently installed. |          |              |          |                          |
| Rcpp   | 1.0.4    | GPL-3.0      | R 3.5.2  |                          |
| The app requires the package Rcpp v1.0.4. Version v1.0.4.6 is currently installed. |          |              |          |                          |
| stringr  | 1.4.0    | GPL-2.0      | TERR 5.0 |                          |
| sva  | 3.30.1   | Artistic-2.0 | R 3.5.2  |                          |
| DMwR   | 0.4.1    | GPL-3.0      | R 3.5.2  |                          |
| caTools  | 1.17.1   | GPL-3.0      | R 3.5.2  |                          |
| parallel   | 5.0.0-66 |              | TERR 5.0 | Java 1.8 x64             |
| The package requires Java v1.8 x64. Version v13.0.2 x64 is currently installed.    |          |              |          |                          |

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

Dependency Manager [Go back to Workflows](#)

**R packages** [Valid Apps](#)

Search...

EXPAND ALL COLLAPSE ALL

**Entity Groups**

| Packages   | Version  | License | Engine   | Non-managed dependencies |
|--|----------|---------|----------|--------------------------|
| parallel   | 5.0.0-66 |         | TERR 5.0 | Java 1.8 x64             |
| This package requires Java v1.8 x64. An incompatible version (v1.7.0.55 x64) is installed but cannot be updated by the dependency manager. |          |         |          |                          |

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

Dependency Manager [Go back to Workflows](#)

**R packages** [Valid Apps](#)

Search...

EXPAND ALL COLLAPSE ALL

**Entity Groups**

| Packages  | Version  | License | Engine   | Non-managed dependencies |
|---|----------|---------|----------|--------------------------|
| parallel  | 5.0.0-66 |         | TERR 5.0 | Java 1.8 x64             |
| This package requires Java v1.8 x64. An incompatible version (v1.8.0.221 x86) is installed but cannot be updated by the dependency manager. |          |         |          |                          |

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

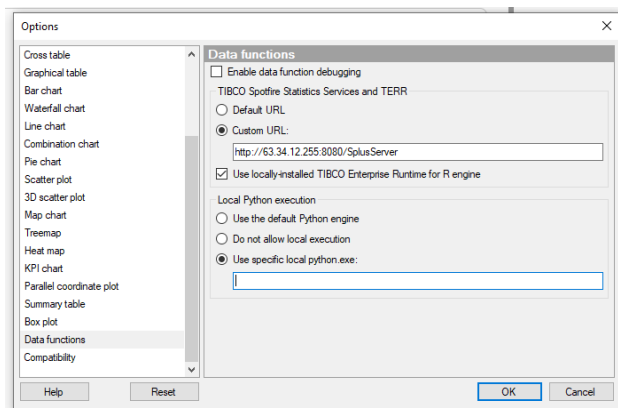
If the TIBCO Spotfire® Data Function option is configured to run data functions in TIBCO Spotfire® Statistics Services, the system administrator must manually install the required dependencies in the TSSS following the installation guide instructions. The *Apps dependency manager* will warn about this situation.

Dependency Manager [Go back to Workflows](#)

This functionality is not available if you are using a TIBCO Spotfire Statistical Services server to run your data functions.

**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 TIBCO 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™ and visible for TSSS. For this run this URL<sup>1</sup> in a web browser<sup>2</sup> with access to the TSSS server:

[http://your-tsss-server:8080/PlusServer/api/v8/expression/eval?cmd=installed.packages\(\)](http://your-tsss-server:8080/PlusServer/api/v8/expression/eval?cmd=installed.packages())

12. An XML with the list of packages installed in TERR™ is available and should include the following packages:

- integromics.hcs
- pki.app.CE.CurveFitting
- pki.app.s4s.CurveFitting
- pki.app.s4s.Normalization
- pki.app.spr.Kinetic

---

<sup>1</sup> This URL requires Expression Service API in the TSSS to be enabled. See [https://docs.tibco.com/pub/sf\\_statsvcs/10.10.0/doc/html/TIB\\_sf\\_statsvcs\\_10.10\\_urlapi/GUID-AB5EA083-5E4D-4D1F-981B-28946A78C5D6.html](https://docs.tibco.com/pub/sf_statsvcs/10.10.0/doc/html/TIB_sf_statsvcs_10.10_urlapi/GUID-AB5EA083-5E4D-4D1F-981B-28946A78C5D6.html) for more information on how to enable this

<sup>2</sup> Microsoft Edge not supported

- pki.app.spr.Preprocessing
- pki.app.spr.RAC
- pki.app.spr.PLA
- signalsDataMerger
- som
- robustbase
- deSolve
- Rcpp
- data.table
- jsonlite
- plyr
- reshape2
- Rtsne
- glue
- magrittr
- stringi
- stringr
- minpack.lm
- PMCMRplus
- PKNCA

### 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 ~ \PlusServer\data\common directory in the TSSS directory structure.

## 3.5 Installation of TERR™ Packages

Signals VitroVivo uses TIBCO® Enterprise Runtime for R (TERR™) as an engine for the analytics. This section explains the steps required to package and deploy all the TERR™ packages used in Signals VitroVivo with the possibility of also including any client TERR™ package.

Before proceeding, ensure you have completed the installation of the TERR™ prerequisites described in the table below and that you have a valid TIBCO Spotfire® user license with administration privileges.

| TIBCO Spotfire® Server                                    | TIBCO® Enterprise Runtime for R (TERR™) |
|---|---|
| <i>TIBCO Spotfire® Server 10.10 + Hotfix012 or latest</i> | <i>TERR™ 6.0</i>                        |
| <i>TIBCO Spotfire® Server 11.4 + Hotfix001 or latest</i>  | <i>TERR™ 6.0</i>                        |

The administrator may choose to either install the TERR libraries without dependencies (see section 3.5.1) or install the TERR libraries to include dependencies (see section 3.5.2). If dependencies are not installed, they must be added using the Dependency Manager prior to launching the Apps, as described in section 3.4.1.

### 3.5.1 Installing Signals VitroVivo TERR™ Libraries (No Dependencies)

To install the *TERR™* packages without dependencies, follow the steps described below:

1. Download and install the *TERR™* described previously according to your *Spotfire® Server* version.
2. Download the Signals VitroVivo *vX.Y.Installers.zip* archive, containing the *TERR\_bundle\_X.Y.Z.build.zip*, from Flexera under the Signals VitroVivo section.
3. Unzip the bundle file *TERR\_bundle\_X.Y.Z.build.zip* into a folder (e.g. in C:\Users\Administrator\Documents).
4. Edit the installation script, *install\_terr\_r\_environments.cmd*, to set the right path to the *TERRScript.exe* present in the previously installed TERR directory (e.g. C:\TERR\terr6.0\bin\TERRscript.exe).

| TIBCO® Enterprise Runtime for R (TERR™) | Installation script                                     |
|---|---|
| <i>TERR™ 6.0</i>                        | <i>TERR\windows\10.10\TERR6.0\generate_terr_spk.cmd</i> |

5. Open a command console and run the installation script *generate\_terr\_spk.cmd* with administrator privileges. The script will install the *TERR™* libraries in the bundle.

```
generate_terr_spk.cmd <package version>
```

E.g.

```
generate_terr_spk.cmd 2.7.0.7500
```

**Note:** Please note that it will take a while to download all the dependencies from CRAN. Ensure the server has internet access and can connect to CRAN (i.e. not blocked by any firewall rule).

6. After 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.
7. A new SPK file named *TIBCO Enterprise Runtime for R Packages Manually\_<version>.spk* is generated.
  - i. Remove “*TIBCO Enterprise Runtime for R Packages <version>*” module from the TIBCO Spotfire® server area. Then validate and save the area.
  - ii. Deploy “*TIBCO Enterprise Runtime for R Packages Manually\_<version>*” module into the TIBCO Spotfire® server area. Then validate and save the area.
  - iii. After deployment, the package “*TIBCO Enterprise Runtime for R Packages Manually <version>*” should appear in the area.

**Note:** The SPK generated is not signed and the client will ask for confirmation when updating the packages.

### 3.5.2 Installing Signals VitroVivo Client TERR™ Libraries and Dependencies

To install the *TERR™* packages to include dependencies please follow the steps described below:

1. Download and install the *TERR™* described previously according to your *TIBCO Spotfire® Server* version.

- Download the Signals VitroVivo vX.Y\_Installers.zip archive, containing the *TERR\_bundle\_X.Y.Z.build.zip*, from Flexera under the Signals VitroVivo section.
- Unzip the bundle file *TERR\_bundle\_X.Y.Z.build.zip* into a folder (e.g. in C:\Users\Administrator\Documents).
- Edit the installation script *install\_terr\_r\_environments.cmd* to set the right path to the *TERRScript.exe* present in the previously installed TERR directory (e.g. C:\TERR\terr6.0\bin\TERRscript.exe).

| TIBCO® Enterprise Runtime for R (TERR™) | Installation script                                     |
|---|---|
| <i>TERR™ 6.0</i>                        | <i>TERR\windows\10.10\TERR6.0\generate_terr_spk.cmd</i> |

- Edit the R script *install\_dependencies.R* with administrator privileges. Add the required dependency libraries to the script in the *terr\_bundle* list (packages with a license allowing distribution) and *terr\_external* (packages with a license that does not allow distribution) list.

| TIBCO® Enterprise Runtime for R (TERR™) | Installation script                                      |
|---|--|
| <i>TERR™ 6.0</i>                        | <i>TERR\windows\10.10\TERR6.0\install_dependencies.R</i> |

```
terr_bundle <- list(
  list(name = 'integromics.hcs',          version = NULL,      server = NULL),
  list(name = 'pki.app.CE.CurveFitting',  version = '1.0.0.0', server = NULL),
  list(name = 'pki.app.s4s.CurveFitting',  version = '1.0.0.0', server = NULL),
  list(name = 'pki.app.s4s.Normalization', version = '1.0.0.0', server = NULL),
  list(name = 'pki.app.spr.Kinetic',       version = '0.9.0.0', server = 'TERR_10.10'),
  list(name = 'pki.app.spr.Preprocessing', version = '1.0.0.0', server = NULL),
  list(name = 'pki.app.spr.RAC',           version = '1.0.0.0', server = NULL),
  list(name = 'pki.app.spr.PLA',           version = '1.0.0.0', server = NULL),
  list(name = 'som',                       version = NULL,      server = NULL),
  list(name = 'signalsDataMerger',        version = NULL,      server = NULL)
);
```

**Note:** Copy the binaries of the packages (zip file) added to the *terr\_bundle* list in the right folder (e.g. *TERR/TERR\_10.10/6.0* folder if *server = TERR\_10.10 + Hotfix-012* or *latest* or *TERR\_11.14 + Hotfix-001* or *latest*).

```
terr_external <- list(
  'jsonlite',
  'plyr',
  'data.table',
  'jsonlite',
  'Rtsne',
  'robustbase',
  'deSolve',
  'stringr',
```

```
'minpack.lm',
'PMCMRplus',
'PKNCA'
);
```

**Note:** It will take a while to download all the dependencies from CRAN. Ensure the server has internet access and can connect to CRAN (i.e. not blocked by any firewall rule).

**Note:** This step may be skipped by manually installing the client packages in *TERR™* using *install.packages()*

6. Edit the R script *generate\_dependencies\_spk.R* with administrative privileges. Add the dependency libraries to the script into the 'lines' variable.

```
lines <- paste("Packages: jsonlite, magrittr, plyr, reshape2, stringi, signalsDataMerger,
Rcpp, stringr, survival, maxstat", "BuiltId: 738ba754-980a-44b5-9560-32b53284d520",
sep='\n')
```

7. Open a command console and run the installation script *generate\_terr\_spk.cmd* with administrator privileges. The script will install the *TERR™* libraries in the bundle.

```
generate_terr_spk.cmd <package version>
```

E.g.

```
generate_terr_spk.cmd 2.7.0.7500
```

**Note:** It will take a while to download all the dependencies from CRAN. Ensure the server has internet access and can connect to CRAN (i.e. is not blocked by any firewall rule).

8. After 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.
9. A new SPK file named *TIBCO Enterprise Runtime for R Packages Manually\_<version>.spk* is generated.
  - i. Remove "*TIBCO Enterprise Runtime for R Packages <version>*" module from the TIBCO Spotfire® server area. Then validate and save the area.
  - ii. Deploy "*TIBCO Enterprise Runtime for R Packages Manually\_<version>*" module into the TIBCO Spotfire® server area. Then validate and save the area.
  - iii. After deployment, the package "*TIBCO Enterprise Runtime for R Packages Manually <version>*" should appear in the area.

**Note:** The SPK generated is not signed and the client will ask for confirmation when updating the packages.

### 3.6 Installation of Signals Inventa

To install Signals Inventa, refer to the *PerkinElmer 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:
  - a. `enableSciStream`: Set this flag to `true` to use the SciStream features

- b. `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 *PerkinElmer 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 TIBCO Spotfire® Extension Packages

To upgrade from earlier versions of Signals VitroVivo (previously called Signals Screening), the new TIBCO 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 3.2

Due to some changes in the package organization in Signals VitroVivo 3.2 there are some additional steps that should be considered when installing the product:

- All products that are part of VitroVivo 3.2 need to be upgraded. This means that Signals Inventa needs to be upgraded to version 3.2 so that the VitroVivo 3.2 Apps work.
- As part of the upgrading process from VitroVivo x.x to VitroVivo 3.2, the customer's administrator should manually remove from the deployment area the following modules:
  - **TIBCO Enterprise Runtime for R Packages**
  - **PerkinElmer Signals Notebook Client**

This does not apply to new installations.

#### 4.1.2 Setting up TIBCO Spotfire® Extension Licenses

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

#### 4.1.3 Signals VitroVivo Apps TIBCO Spotfire® Extension Configuration

Review the configuration of the TIBCO 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.

#### 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 stop the Metastore Service as a daemon process as admin:

```
sudo docker-compose stop
```

6. 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
```

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

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

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 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 Error! Hyperlink reference not valid.
```

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.

### 4.3.2 Upgrading the Signals VitroVivo Metastore Service on a Windows Machine

Follow the next steps to upgrade your Metastore Service:

1. Ensure existing data is backed up.
2. Extract the `docker-compose-<OS>-<version>.tar.gz` with 7zip in the desired location in the file system of the Metastore service host.
3. (Optional) Edit `.env` file with a text editor, modifying the `METASOTRE_PORT` if configuring a different TCP/IP port than the default one.
4. 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:

```
docker-compose stop
```

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

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

6. Execute the following shell command to check that the right Docker version is installed, the following string ``NAMES`` is `"pkiinformatics/sss-metastore:<version>"` is displayed:

```
docker container ls -a
```



7. 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 that you have specified in the `.env` file.

You can also 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 Error! Hyperlink reference not valid.
```

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:

- 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>.

- 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
```

- 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
```

- 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 attempting to paste a large amount of data into the Web Player using the Editable Data Grid App, an error may be raised if the Web Player exceeds a maximum size of 256kb for request and response messages sent to and from the service. This issue can be solved by increasing the maximum size limit of the messages that the Web Player accepts. To do this, follow the procedure in:

<http://informatics.perkinelmer.com/Support/KnowledgeBase/details/Default?TechNote=8693>

## 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:

- The 64-bit version for JRE version 8.0 or higher is installed.
- 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**).
- Add JRE bin folder to Path environment variable.
- Follow the indications to set JAVA\_HOME from TERR™ available at: [https://docs.tibco.com/pub/enterprise-runtime-for-R/4.2.0/doc/html/TIB\\_terr\\_Package-Management/GUID-2559E306-F98C-4FB9-8B3C-B16914888A30.html](https://docs.tibco.com/pub/enterprise-runtime-for-R/4.2.0/doc/html/TIB_terr_Package-Management/GUID-2559E306-F98C-4FB9-8B3C-B16914888A30.html).

## 6 Backups

To mitigate potential damage due to hardware or software failure, it is highly recommend 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 TIBCO 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 TIBCO 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 TIBCO 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 TIBCO 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 TIBCO 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 TIBCO 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`

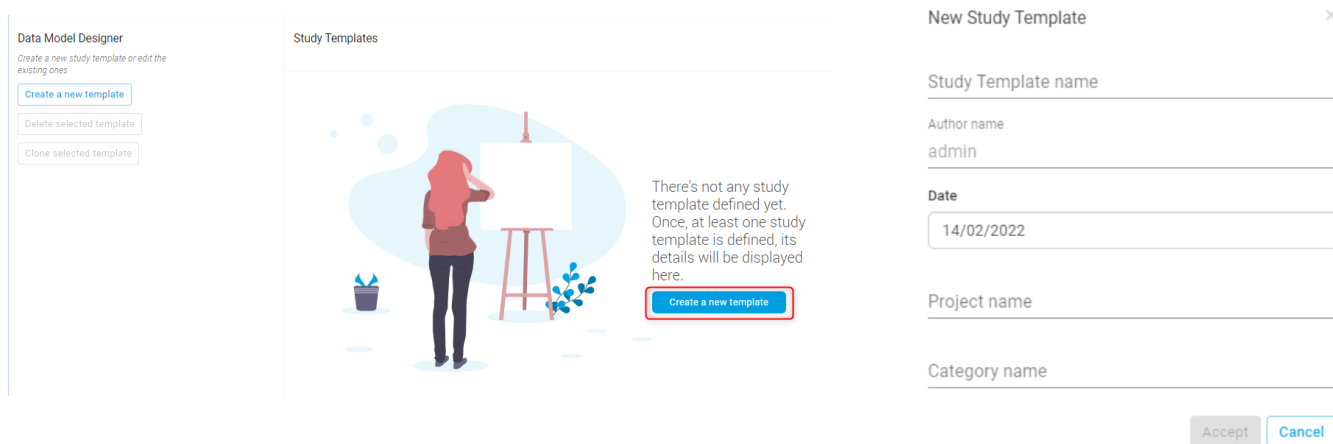
## 7 Appendices

### 7.1 Appendix In Vivo Data Model Designer App

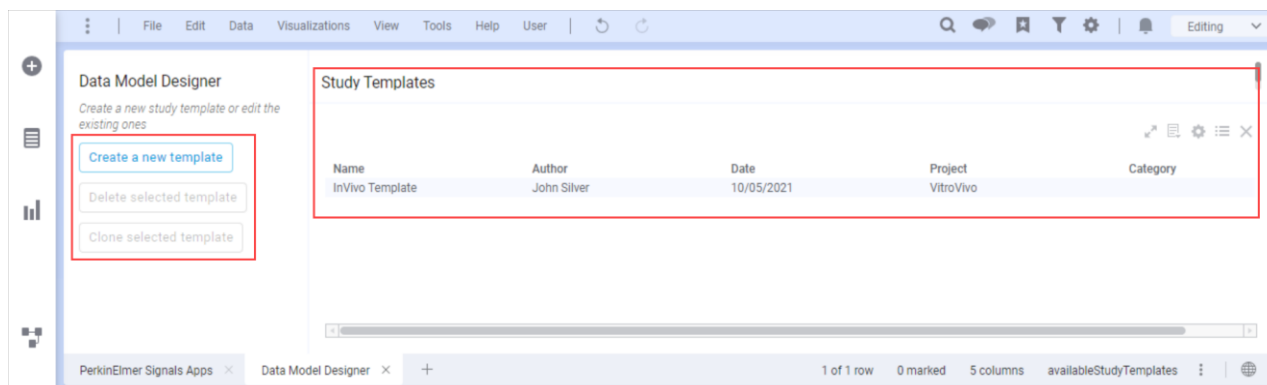
Using the **Data Model Designer** App, the administrator may configure and edit the In Vivo metadata and Administrator Defined Tables (ADTs) of a Study Template using data grids. The App will notify when no Study Templates are defined, suggesting the creation of a new one by selecting any of the **Create a new template** buttons and filling in the resulting form with Study Template name (required), author (auto-populated, not editable), date, project name and category (optional fields).



*Data Model Designer App Card*

The screenshot shows the Data Model Designer application interface. On the left, there is a sidebar with the title "Data Model Designer" and the subtitle "Create a new study template or edit the existing ones". Below this, there are three buttons: "Create a new template" (highlighted in blue), "Delete selected template", and "Clone selected template". The main area is titled "Study Templates" and contains an illustration of a person standing next to a whiteboard. A text box next to the illustration says: "There's not any study template defined yet. Once, at least one study template is defined, its details will be displayed here." Below this text is a red-bordered button labeled "Create a new template". On the right side of the interface, there is a "New Study Template" dialog box with a close button (X). It contains several input fields: "Study Template name", "Author name" (with the value "admin"), "Date" (with the value "14/02/2022"), "Project name", and "Category name". At the bottom of the dialog are "Accept" and "Cancel" buttons.

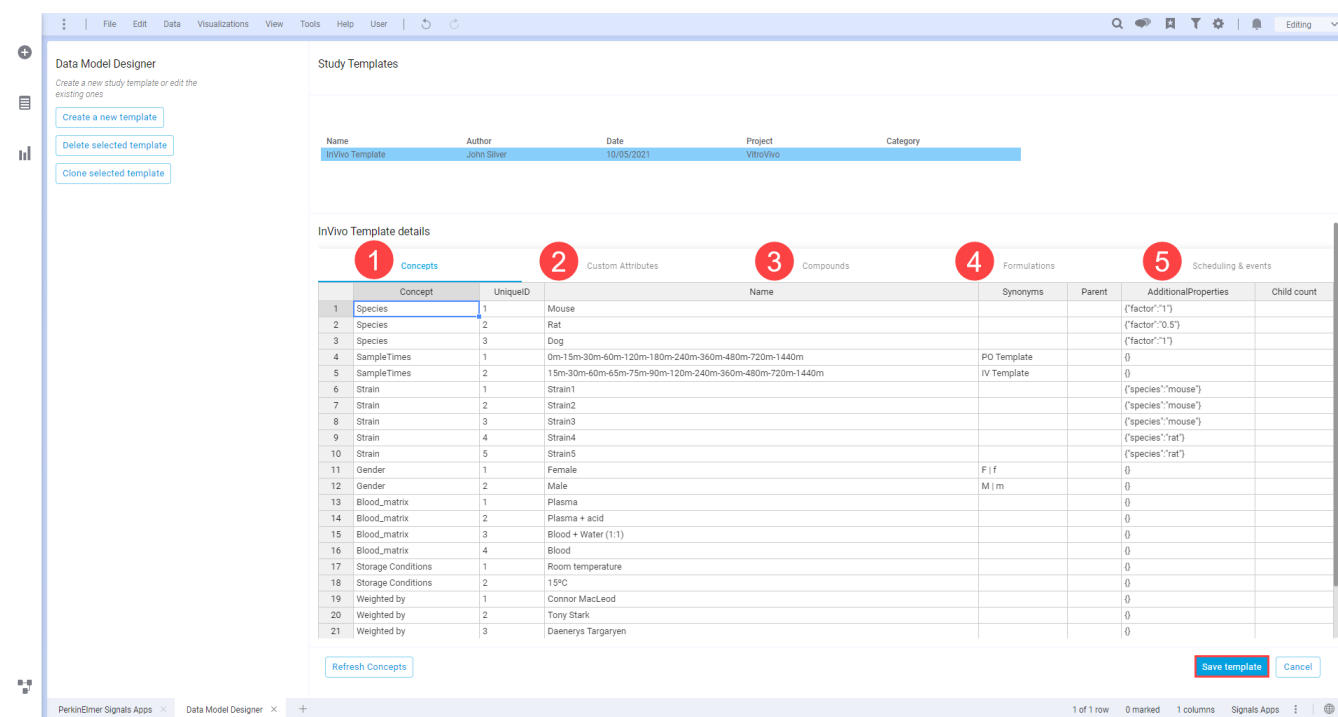
Existing Study Templates (if applicable) will be listed in the **Study Templates** section of the DMD App in the right-hand panel. Selecting a Study template to automatically load and edit it. A selected Study Template can be deleted or cloned using the corresponding buttons in the left-hand panel. Users can create a new Study Template at any time by selecting the **Create a new template** button.



The Study Template edition view contains **five** sections:

1. Concepts
2. Custom Attributes
3. Compounds
4. Formulations
5. Scheduling & events

When the **Save template** button is selected, the concepts, custom attributes, compounds, formulations and scheduling & events are saved to the corresponding Study Template subfolder, inside the folder created and configured previously as a preference.



## 7.1.1 Concepts

**Concepts** are controlled vocabulary that can be used to populate **Concept Attributes** and are visualized as a list of options. Each row of the grid is an option, and all the options with the same Concept name, will be grouped as a Concept. The most relevant columns are:

1. **Concept:** the name of the concept. Use this value in the 'Concept' column of the Define Study attributes section (e.g. "Gender").
2. **UniqueID:** each option for each concept should have a unique ID and it is recommended to use incremental numeric IDs.
3. **Name:** the value of the concept option (e.g. "Female").
4. **Synonyms:** this column is used to configure the display name of the core Concept (e.g. "SampleTimes").
5. **AdditionalProperties:** a json formatted string used to specify the Species Factor of the defined species. This value is used by the Dose Preparation App when generating recipes. In the case of strains, this column can be used to indicate the species to which they are linked.

The remaining columns exist for integration purposes.

| Concepts      |          | Custom Attributes                                      | Compounds   | Formulations | Scheduling & events  |             |
|---------------|----------|--|-------------|--------------|----------------------|-------------|
| Concept       | UniqueID | Name   | Synonyms    | Parent       | AdditionalProperties | Child count |
| 1 Species     | 1        | Mouse  |             |              | {"factor":1}         |             |
| 2 Species     | 2        | Rat  |             |              | {"factor":0.5}       |             |
| 3 Species     | 3        | Dog  |             |              | {"factor":1}         |             |
| 4 SampleTimes | 1        | 0m-15m-30m-60m-120m-180m-240m-360m-480m-720m-1440m     | PO Template |              | {}                   |             |
| 5 SampleTimes | 2        | 15m-30m-60m-65m-75m-90m-120m-240m-360m-480m-720m-1440m | IV Template |              | {}                   |             |
| 6 Strain      | 1        | Strain1  |             |              | {"species":"mouse"}  |             |
| 7 Strain      | 2        | Strain2  |             |              | {"species":"mouse"}  |             |
| 8 Strain      | 3        | Strain3  |             |              | {"species":"mouse"}  |             |
| 9 Strain      | 4        | Strain4  |             |              | {"species":"rat"}    |             |
| 10 Strain     | 5        | Strain5  |             |              | {"species":"rat"}    |             |
| 11 Gender     | 1        | Female   | F   f       |              | {}                   |             |
| 12 Gender     | 2        | Male   | M   m       |              | {}                   |             |

1 Refresh Concepts      2      3      4      5 Save template Cancel

**Note:** Defined Concepts are only available (e.g. to define a new Study Attribute) after the **Refresh Concepts** button is selected.

**Note:** 'Species', 'SampleTimes', 'Strain' and 'Gender' Concepts are mandatory. At least one option/value for each mandatory Concept must be defined. 'Route' is a reserved concept and cannot be defined by users.

## 7.1.2 Custom Attributes

This section can be used to extend and customize fields in any supporting In Vivo App. The most relevant columns are:

1. **Name:** the display name of the custom field
2. **Area:** the In Vivo App area in which the custom field is displayed. The available areas are:
  - Study Definition
  - Treatment
  - Stock Solutions
  - Dose Preparation
  - Baseline

3. **DataType**: the type of input field. The supported types are:
  - **Text**: single line input field
  - **Textarea**: multi line input field
  - **Date**: date picker component
  - **Concept**: a selector with controller vocabulary defined in the **Define Concepts** section. Only already defined concepts will be available.
4. **Concept**: specify a concept name to configure the controlled vocabulary of the attribute with 'Concept' DataType
5. **IsRequired**: set this value to TRUE if the field is required and FALSE if it is optional
6. **DefaultValue**: a value manually introduced by the user that will be displayed by default in the corresponding field
7. **Settings**: json formatted string for additional configuration

The remaining columns exist for integration purposes.

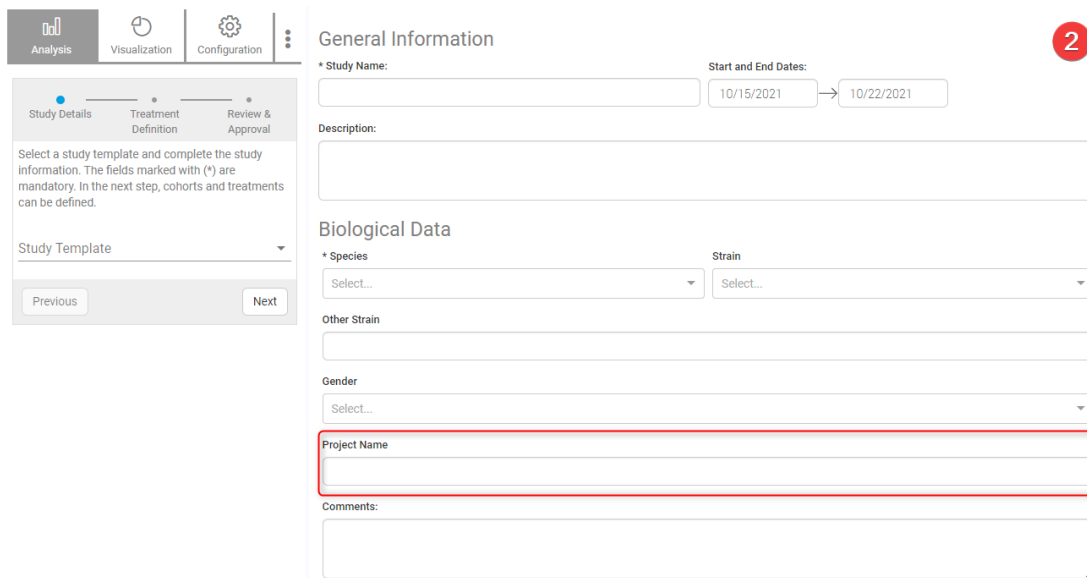
| Concepts |  | Custom Attributes |          |            | Compounds          |        |            | Formulations |                  | Scheduling & events                    |
|----------|--|-------------------|----------|------------|--------------------|--------|------------|--------------|------------------|--|
|          | Name                                     | Area              | DataType | ColumnType | Concept            | IsCore | IsRequired | Route        | DefaultValue     | Settings                               |
| 1        | Project ID                               | Study Definition  | Text     | String     |                    | False  | False      |              |                  | "required":false,"group":"","rank":1)  |
| 2        | Study Number                             | Study Definition  | Text     | String     |                    | False  | False      |              |                  | "required":false,"group":"","rank":20) |
| 3        | Program Name                             | Study Definition  | Text     | String     |                    | False  | False      |              |                  | "required":false,"group":"","rank":30) |
| 4        | PACT                                     | Study Definition  | Text     | String     |                    | False  | False      |              |                  | "required":false,"group":"","rank":43) |
| 5        | Number of Samples                        | Study Definition  | Text     | String     |                    | False  | False      |              |                  | "required":false,"group":"","rank":42) |
| 6        | Notebook Number                          | Study Definition  | Text     | String     |                    | False  | False      |              |                  | "required":false,"group":"","rank":41) |
| 7        | Matrix                                   | Study Definition  | Concept  | String     | Blood_matrix       | False  | False      |              |                  | "required":false,"group":"","rank":40) |
| 8        | Formulation Procedure                    | Dose Preparation  | Textarea | String     |                    | False  | False      |              |                  | "required":false,"group":"","rank":0)  |
| 9        | Record final appearance after filtration | Dose Preparation  | Text     | String     |                    | False  | False      | PO           |                  | "required":false,"group":"","rank":0)  |
| 10       | Record final appearance                  | Dose Preparation  | Text     | String     |                    | False  | False      | PO           |                  | "required":false,"group":"","rank":0)  |
| 11       | Measure final pH                         | Dose Preparation  | Text     | Real       |                    | False  | False      |              |                  | "required":false,"group":"","rank":0)  |
| 12       | Method used for determination of pH      | Dose Preparation  | Text     | String     |                    | False  | False      |              |                  | "required":false,"group":"","rank":0)  |
| 13       | Stir Overnight                           | Dose Preparation  | Text     | String     |                    | False  | False      |              |                  | "required":false,"group":"","rank":0)  |
| 14       | Storage Conditions                       | Dose Preparation  | Concept  | String     | Storage Conditions | False  | False      |              | Room temperature | "required":false,"group":"","rank":0)  |
| 15       | Comments                                 | Dose Preparation  | Textarea | String     |                    | False  | False      |              |                  | "required":false,"group":"","rank":0)  |
| 16       | Prepared by                              | Dose Preparation  | Text     | String     |                    | False  | False      |              |                  | "required":false,"group":"","rank":0)  |
| 17       | Checked by                               | Dose Preparation  | Text     | String     |                    | False  | False      |              |                  | "required":false,"group":"","rank":0)  |
| 18       | Weighted by                              | Stock Solutions   | Concept  | String     | Weighted by        | False  | False      |              |                  | "required":false,"group":"","rank":0)  |
| 19       | Designed by                              | Treatment         | Concept  | String     | Weighted by        | False  | False      |              |                  | "required":false,"group":"","rank":0)  |
| 20       | Additional Information                   | Stock Solutions   | Text     | String     |                    | False  | False      |              |                  | "required":false,"group":"","rank":0)  |
| 21       | Head Size                                | Baseline          | Text     | Real       |                    | False  | False      |              |                  | "required":false,"group":"","rank":0)  |

For example, if a custom attribute named 'Project Name' is defined in a Study Template named 'Study Template Example' **1**, the field will be prompted in the corresponding Workflow step when that Study Template is selected in the **Study Designer App** **2**.

Study Template Example details **1**

| Concepts |              | Custom Attributes |          |            | Compounds |        |            | Formulations |              | Scheduling & events                    |
|----------|--------------|-------------------|----------|------------|-----------|--------|------------|--------------|--------------|--|
|          | Name         | Area              | DataType | ColumnType | Concept   | IsCore | IsRequired | Route        | DefaultValue | Settings                               |
| 1        | Project Name | Study Definition  | Text     | String     |           | False  | False      |              |              | ("required":false,"group":"","rank":0) |
| 2        |              |                   |          |            |           |        |            |              |              |  |





**General Information**

\* Study Name:  Start and End Dates: 10/15/2021 → 10/22/2021

Description:

**Biological Data**

\* Species:  Strain:

Other Strain:

Gender:

Project Name:

Comments:

### 7.1.3 Compounds

The administrator can define the compounds to be loaded in the *Treatment Definition* step in the **Study Designer** App. The most relevant columns are:

1. **Compound ID**: unique identifier of the compound
2. **Parent Molecular Weight**: numeric weight value
3. **Lot Id**: lot Id reference
4. **BEW**: bio equivalent weight

The remaining columns exist for integration purposes.

**Note:** Compounds are considered mandatory. At least one must be defined.

|   | Concepts    | Custom Attributes       | Compounds      | Formulations | Scheduling & events |          |
|---|-------------|-------------------------|----------------|--------------|---------------------|----------|
|   | Compound ID | Parent Molecular Weight | Lot Id         | BEW          | Version             | Comments |
| 1 | 5163638A    | 416.412                 | REG00479224-01 | 1            |                     |          |
| 2 | 5218131A    | 384.351                 | REG00479138-01 | 1            |                     |          |
| 3 | 5163638B    | 439.258                 | REG00479224-02 | 1            |                     |          |
| 4 | 5218131B    | 333.333                 | REG00479138-02 | 1            |                     |          |
| 5 |             |                         |                |              |                     |          |

### 7.1.4 Formulations

Formulations are composed of a list of elements. Each row of the grid is an element present in a specific formulation, and all the elements with the same Formulation ID will be grouped to define a Formulation. In the example screenshot below two formulations are defined:

- **F6**: 10% Cavitron + 20% DMSO
- **F7**: 10% Solutol + 10% DMA


The most relevant columns are:

1. **FormulationID**: the ID of the formulation
2. **Percentage**: percentage of the formulation element
3. **Tag**: name of the formulation element

**Note:** Formulations are considered mandatory. At least one must be defined.

|   | Concepts | Custom Attributes | Compounds | Formulations   | Scheduling & events |          |
|---|----------|-------------------|-----------|----------------|---------------------|----------|
|   |          |                   |           | Formulation ID | Percentage          | Tag      |
| 1 | F6       |                   |           | 10             |                     | Cavitron |
| 2 | F6       |                   |           | 20             |                     | DMSO     |
| 3 | F7       |                   |           | 10             |                     | Solutol  |
| 4 | F7       |                   |           | 10             |                     | DMA      |
| 5 |          |                   |           |                |                     |          |

### 7.1.5 Scheduling & Events

By default, three data grids representing the basic event types (Baselining, Dosing, and Sampling) are displayed and can be edited directly or renamed or deleted using the  button on the right-hand side. Users may add a new event type by selecting the **Create new event** button. Users may add custom attributes for each event type from this page using the data grid. The relevant columns are:

1. **Name**: the display name of the custom event type attribute
2. **DataType**: the type of the input field. The supported types are:
  - **Text**: single line input field
  - **Textarea**: multi line input field
  - **Date**: date picker component
  - **Concept**: a selector with controller vocabulary defined in the **Define Concepts** section
  - **Integer**: integer number input field
  - **Real**: real number input field
  - **Boolean**: TRUE/FALSE selector
3. **Concept**: specify a concept name to configure the controlled vocabulary of the attribute with 'Concept' DataType
4. **DefaultValue**: a value manually introduced by the user that will be displayed by default in the corresponding field

InVivo Template details

Concepts Custom Attributes Compounds Formulations Scheduling & events

[Create new event](#)

Baselining *Baselining*

|   | Name | DataType | Concept | DefaultValue |
|---|------|----------|---------|--------------|
| 1 |      |          |         |              |

Dosing *Dosing*

|   | Name     | DataType      | Concept | DefaultValue |
|---|----------|---------------|---------|--------------|
| 1 | Reviewer | Concept       |         |              |
| 2 |          | Invoignted by |         |              |

[Refresh Concepts](#) [Save template](#) [Cancel](#)

## 7.2 Appendix In Vivo Integration with Signals Notebook

To fully utilize the features available in the In Vivo Apps, a connection to Signals Notebook is required. Signals Notebook is used to store experiment information to be shared with different users in support of concurrent

editing, and additionally can supply a compound table (see the In Vivo section of the *PerkinElmer Signals VitroVivo User Guide* for details on this feature).

When defining a **Study Template** using the **Data Model Designer** App, the corresponding experiment events must be defined, where at least one event representing each of the basic event metatypes: *Baselining*, *Dosing*, and *Sampling*, is required. The corresponding tables of these defined events must first exist inside a **Signals Notebook Experiment** to be populated by the following In Vivo Apps:

- Study Designer
- Baseline Capture
- Sequence of Events

It is recommended that the Signals Notebook Administrator creates an Admin Defined Table (ADT) template for each of their **Study Template events** so that they may be quickly added and appended to (when required) for inclusion in a Signals Notebook experiment by a standard user.

### 7.2.1 Creating Admin Define Tables (ADTs) in Signals Notebook

There are three different categories of columns in each event type ADT:

1. **Mandatory columns** – consistent across all event types
2. **Metatype specific columns** – the core set of attributes specific to the event type
3. **Custom columns** – additional columns defined by the user in the **Data Model Designer** App

| Mandatory |        |           |            |                    |                          |                   |                         |                  |                        |                 |                       |       | Metatype Specific  |                               |                                | Custom                  |          |            |          |
|-----------|--------|-----------|------------|--------------------|--------------------------|-------------------|-------------------------|------------------|------------------------|-----------------|-----------------------|-------|--------------------|-------------------------------|--------------------------------|-------------------------|----------|------------|----------|
| EventID   | Cohort | Animal ID | Event Type | Start Nominal Time | Start Nominal Clock Time | Start Actual Time | Start Actual Clock Time | End Nominal Time | End Nominal Clock Time | End Actual Time | End Actual Clock Time | State | Time Infused (min) | Pre-Dosing Syringe Weight (g) | Post-Dosing Syringe Weight (g) | Manual Theoretical Dose | Dosed by | Tube Color | Comments |
| 3         | A      | Rat 1     | Dosing     | 0m                 | 2022-06-09 3:07 PM       | 0m                | 2022-06-09 3:07 PM      |                  |                        |                 |                       | Done  | 60                 | 7.795                         | 6.271                          |                         |          |            |          |
| 4         | A      | Rat 2     | Dosing     | 0m                 | 2022-06-09 3:07 PM       | 0m                | 2022-06-09 3:07 PM      |                  |                        |                 |                       | Done  | 60                 | 7.955                         | 6.299                          |                         |          |            |          |
| 31        | B      | Rat 3     | Dosing     | 0m                 | 2022-06-09 3:07 PM       | 0m                | 2022-06-09 3:07 PM      |                  |                        |                 |                       | Done  | 60                 | 10.061                        | 6.406                          |                         |          |            |          |
| 32        | B      | Rat 4     | Dosing     | 0m                 | 2022-06-09 3:07 PM       | 0m                | 2022-06-09 3:07 PM      |                  |                        |                 |                       | Done  | 60                 | 10.038                        | 6.395                          |                         |          |            |          |

**Note:** The naming convention for the ADT tables and columns are case sensitive and must appear exactly as defined in the corresponding **Study Template** to ensure they correctly link between Signals Notebook and VitroVivo. To view the expected ADT columns for an event type, open the **Data Model Designer** App, select the **Scheduling & events** tab > icon next to the desired event > **Signals Notebook ADT Info**. A dialog with the expected ADT columns will be displayed.

Signals Notebook ADT Info

This is the expected Signals Notebook ADT format

ADT Name:



ADT Columns:

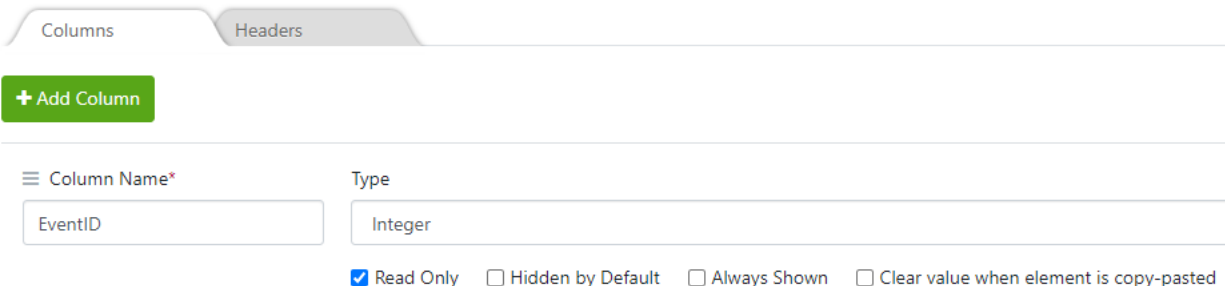
- EventID
- Animal ID
- Cohort
- Event Type
- Start Nominal Time
- Start Nominal Clock Time
- Start Actual Time
- Start Actual Clock Time
- End Nominal Time
- End Nominal Clock Time
- End Actual Time
- End Actual Clock Time
- State
- Animal Re-use
- Tickatlab Info
- Body Weight (kg)
- Culex Position
- Route
- Dose Volume (mL/kg)
- Infusion Length (m)
- Comments

ADT Name and Columns correspond to the currently stored template information. Please, click on the Save button to apply any change and refresh the information.

#### Signals Notebook ADT Info for the Baselining Events ADT

### To create an ADT template in Signals Notebook:

1. From Signals Notebook, select the ▼ dropdown arrow next to your username > **System Configuration**.
2. Log in to Signals Notebook using credentials with administrative privileges.
3. Select the **Table Template**  icon from the main toolbar.
4. Select the **Table Template**  button.
5. Give the template a name > **Create Table**.
6. Add a column by giving it a name > choose the appropriate **Type** > + **Add Column**. Note that **EventID** must be set as **Type 'Integer'**. All other columns must be set as **Type 'Text'**.



Columns Headers

+ Add Column

Column Name\* Type

EventID Integer

Read Only  Hidden by Default  Always Shown  Clear value when element is copy-pasted

7. Add each of the **mandatory columns** by naming them as shown below. Note all columns should be marked as **'Read Only'**.

| Mandatory Column         | Type    |
|--------------------------|---------|
| EventID                  | Integer |
| Cohort                   | Text    |
| Animal ID                | Text    |
| Event Type               | Text    |
| Start Nominal Time       | Text    |
| Start Nominal Clock Time | Text    |
| Start Actual Time        | Text    |
| Start Actual Clock Time  | Text    |
| End Nominal Time         | Text    |
| End Nominal Clock Time   | Text    |
| End Actual Time          | Text    |
| End Actual Clock Time    | Text    |
| State                    | Text    |

8. A preview of the template is shown in the upper-left hand side of the screen, as shown below.

Preview [Test It Out](#)

| EventID | Cohort    | Animal ID | Event Type | Start Nominal Time | Start Nominal Clock Time | Start Actual Time | Start Actual Clock Time | End Nominal Time | End Nominal Clock Time | End Actual Time | End Actual Clock Time | State     |
|---------|-----------|-----------|------------|--------------------|--------------------------|-------------------|-------------------------|------------------|------------------------|-----------------|-----------------------|-----------|
| 1       | Some text | Some text | Some text  | Some text          | Some text                | Some text         | Some text               | Some text        | Some text              | Some text       | Some text             | Some text |

[Cancel](#) [Save](#)

9. To add **metatype specific columns** based on event type, see the appropriate section below. Alternatively, a user may add these columns later, as described in the section, *Adding an ADT to a Signals Notebook Experiment*.
10. Select **Save**.

### 7.2.1.1 Baseline Events ADT

In addition to the **mandatory columns** described in *Creating Admin Define Tables (ADTs)* in Signals Notebook, add the following columns specific to the basic baselining events ADT table. Note all columns should be marked as **'Read Only'** and **Type 'Text'**.

#### Metatype Columns:

1. Animal Re-use
2. Tickatlab Info
3. Body Weight (kg)
4. Culex Position
5. Route
6. Dose Volume (mL/kg)
7. Infusion Length (m)
8. Comments

**Note:** The 'Comments' column always appears as the last column of the ADT. Therefore, custom columns added by the user in the **Study Template** will appear before this column.

### 7.2.1.2 Dosing Events ADT

In addition to the **mandatory columns** described in *Creating Admin Define Tables (ADTs)* in Signals Notebook, add the following columns specific to the basic dosing events. Note all columns should be marked as **'Read Only'** and **Type 'Text'**.

#### Metatype Columns:

1. Time Infused (min)
2. Pre-Dosing Syringe Weight (g)
3. Post-Dosing Syringe Weight (g)
4. Manual Theoretical Dose
5. Dosed by
6. Comments

**Note:** The 'Comments' column always appears as the last column of the ADT. Therefore, custom columns added by the user in the **Study Template** will appear before this column.

### 7.2.1.3 Sampling Events ADT

In addition to the **mandatory columns** described in *Creating Admin Define Tables (ADTs)* in Signals Notebook, add the following columns specific to the basic sampling events. Note all columns should be marked as **'Read Only'** and **Type 'Text'**.

#### Metatype Columns:

1. Comments

**Note:** The 'Comments' column always appears as the last column of the ADT. Therefore, custom columns added by the user in the **Study Template** will appear before this column.

## 7.2.2 Adding an ADT to a Signals Notebook Experiment

To add an ADT template to a new experiment in Signals Notebook:

1. Select **Add New > Experiment** from the top-right hand side of the main toolbar in Signals Notebook.
2. Provide a **Name** for the experiment and any other optional details.
3. Select **Create Experiment**.
4. Select **Add Content > Table**.
5. From the **Table Template** dropdown, select the desired event type ADT template > **Create**.
6. Continue to add ADTs until a table exists for each of the experiment's event types. Note these are defined in the **Study Template** using the **Data Model Designer** App.
7. Double-click on each table name in the left-hand panel and rename to match the name specified in the **Scheduling & events** tab of the appropriate **Study Template** using the format '[event name] Events'. The naming convention for the ADT tables are case sensitive and must match to correctly link between Signals Notebook and VitroVivo. An example is shown below:

| Event Name in VitroVivo | Event Name in Signals Notebook |
|-------------------------|--------------------------------|
| 'Baselining'            | 'Baselining Events'            |
| 'Dosing'                | 'Dosing Events'                |
| 'Sampling'              | 'Sampling Events'              |

NonSensitiveInVivoStdAttributes details

Concepts   Custom Attributes   Compounds   Formulations   **Scheduling & events**

Create new event

Baselining *Baselining*

| Name | Data Type | Concept | DefaultValue |
|------|-----------|---------|--------------|
| 1    |           |         |              |

Dosing *Dosing*

| Name | Data Type | Concept | DefaultValue |
|------|-----------|---------|--------------|
| 1    |           |         |              |

Sampling *Sampling*

| Name | Data Type | Concept | DefaultValue |
|------|-----------|---------|--------------|
| 1    |           |         |              |

*Event types as displayed in VitroVivo*

Signals Notebook   Quick Find   Add New   0   Katelyn Moriarty

nb1 > InVivo Experiment for Doc   No description   ACTIVE   SHARED   ChemACK

Contents   Comments   Add Content   Properties   History   Status   Related

Experiment Contents

Content

- Baselining Events
- Dosing Events
- Sampling Events

Properties

Baselining Events

Row Count: 4 (of 2000 limit)

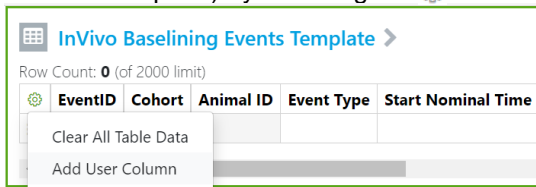
| EventID | Cohort | Animal ID | Event Type | Start Nominal Time | Start Nominal Clock Time | Start Actual Time | Start Actual Clock Time |
|---------|--------|-----------|------------|--------------------|--------------------------|-------------------|-------------------------|
| 1       | A      | Rat 1     | Baselining | 0m                 |                          | 0m                |                         |
| 2       | A      | Rat 2     | Baselining | 0m                 |                          | 0m                |                         |
| 29      | B      | Rat 3     | Baselining | 0m                 |                          | 0m                |                         |
| 30      | B      | Rat 4     | Baselining | 0m                 |                          | 0m                |                         |

Dosing Events

| EventID | Cohort | Animal ID | Event Type | Start Nominal Time | Start Nominal Clock Time | Start Actual Time | Start Actual Clock Time |
|---------|--------|-----------|------------|--------------------|--------------------------|-------------------|-------------------------|
|         |        |           |            |                    |                          |                   |                         |

*Event type tables renamed in Signals Notebook*

8. Add additional **custom columns** or **metatype specific columns** (if not included by the administrator in the ADT template) by selecting the  icon > **Add User Column**.

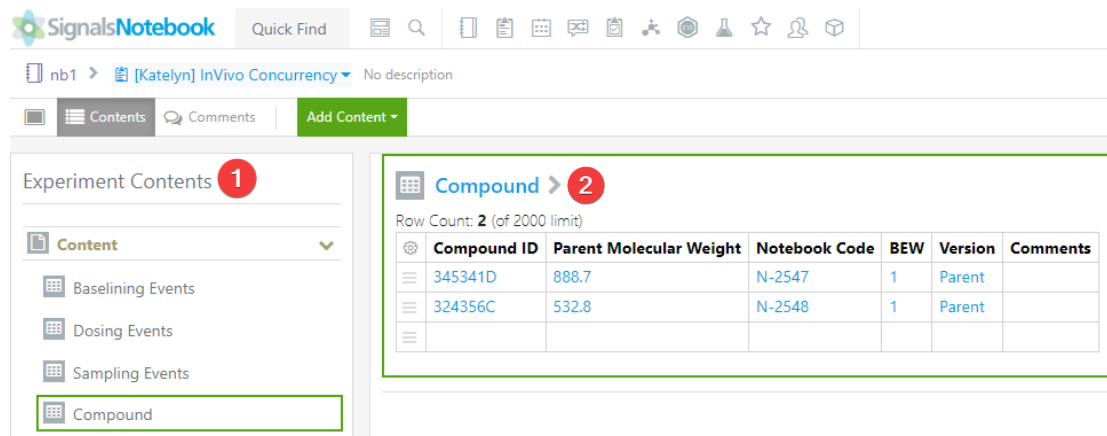


9. The column will be added to the end of the table, rename it by selecting the 'User Column' heading > **Rename Column**.
10. Click and drag the column to the desired position in the table to reorder. Note that columns do not need to be in the exact order as displayed in the **Study Template**, however it is recommended.

### 7.2.3 Adding a Compound Table to a Signals Notebook Experiment

The administrator will need to add entries to a Signals Notebook Experiment and configure the In Vivo Workflow to download compounds from Signals Notebook in the **Study Designer** App. To add a compound table to Signals Notebook:

- i. Log in to Signals Notebook.
- ii. Select the desired experiment, as shown in **1** below.
- iii. Add a table with compound information, as shown in **2**. The table layout requires one compound per row and must have at least four columns with information containing: **Compound ID**, **Lot ID**, **Molecular Weight**, and **BEW**. The name of the compounds table and its column titles are insignificant as they will be configured in a later step by the **Study Designer** App.



| Compound ID | Parent Molecular Weight | Notebook Code | BEW | Version | Comments |
|-------------|-------------------------|---------------|-----|---------|----------|
| 345341D     | 888.7                   | N-2547        | 1   | Parent  |          |
| 324356C     | 532.8                   | N-2548        | 1   | Parent  |          |

## 7.3 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

### 7.3.1 Web Player System Requirements

| Category                 | Requirement   |
|--------------------------|---|
| <b>Hardware</b>          |   |
| Processor                | <ul style="list-style-type: none"> <li>• Minimum: 2 Cores, 2 GHz</li> <li>• Recommended: 4 Cores or more (Intel Core i5 or equivalent), 2+ GHz, 64-bit</li> </ul>   |
| RAM                      | <ul style="list-style-type: none"> <li>• Minimum: 8 GB</li> <li>• Recommended: 16 GB or greater</li> </ul> <p>Large datasets can require more RAM.</p>  |
| Hard Disk Space          | <ul style="list-style-type: none"> <li>• 10 GB is recommended for installation and normal use.</li> </ul>   |
| Display                  | <ul style="list-style-type: none"> <li>• Minimum: 1024x768 pixel resolution, 16 or 32-bit color depth.</li> <li>• Recommended: 1366x768 pixel resolution or higher, 16 or 32-bit color depth.</li> </ul> <p>For functionality that require hardware acceleration (such as the 3D plot), DirectX 9 or higher and a compatible graphics card is required</p>  |
| <b>Software</b>          |   |
| Operating System         | <ul style="list-style-type: none"> <li>• Windows 10 64-bit</li> </ul>   |
| TIBCO Spotfire® Software | <ul style="list-style-type: none"> <li>• TIBCO Spotfire® Web Player 10.10.x server, see <a href="https://docs.tibco.com/pub/spotfire/general/sr/sr/topics/spotfire_web_player_10_10_lts.html">https://docs.tibco.com/pub/spotfire/general/sr/sr/topics/spotfire_web_player_10_10_lts.html</a> for TIBCO Spotfire® Web Player system requirements</li> <li>• TIBCO Spotfire® Web Player 11.4.x server, see <a href="https://docs.tibco.com/pub/spotfire/general/sr/sr/topics/spotfire_web_player_11_4_lts.html">https://docs.tibco.com/pub/spotfire/general/sr/sr/topics/spotfire_web_player_11_4_lts.html</a> for TIBCO Spotfire® Web Player system requirements</li> <li>• TIBCO Spotfire® Web Player 12.0.x server, see <a href="https://docs.tibco.com/pub/spotfire/general/sr/sr/topics/spotfire_web_player_12_0.html">https://docs.tibco.com/pub/spotfire/general/sr/sr/topics/spotfire_web_player_12_0.html</a> for TIBCO Spotfire® Web Player system requirements</li> </ul> |



### 7.3.2 TSSS System Requirements

| Category                             | Requirement   |
|--------------------------------------|---|
| <b>Hardware</b>                      |   |
| Processor                            | <ul style="list-style-type: none"> <li>• Minimum: 2 Cores, 2 GHz</li> <li>• Recommended: 4 Cores or more (Intel Core i5 or equivalent), 2+ GHz, 64-bit</li> </ul>   |
| RAM                                  | <ul style="list-style-type: none"> <li>• Minimum: 8 GB</li> <li>• Recommended: 16 GB or greater</li> </ul> <p>Large datasets can require more RAM.</p>  |
| Hard Disk Space                      | <ul style="list-style-type: none"> <li>• 10 GB is recommended for installation and normal use.</li> </ul>   |
| Display                              | <ul style="list-style-type: none"> <li>• Minimum: 1024x768 pixel resolution, 16 or 32-bit color depth.</li> <li>• Recommended: 1366x768 pixel resolution or higher, 16 or 32-bit color depth.</li> </ul> <p>For functionality that requires hardware acceleration (such as the 3D plot), DirectX 9 or higher a compatible graphics card is required.</p>  |
| <b>Software</b>                      |   |
| Operating System                     | <ul style="list-style-type: none"> <li>• Windows 10 64-bit</li> </ul>   |
| TIBCO Spotfire® Statistical Services | <ul style="list-style-type: none"> <li>• TIBCO Spotfire® Statistical Services 10.10.x server, see <a href="https://docs.tibco.com/pub/sf_statsvcs/10.10.0/doc/html/TIB_sf_statsvcs_sys-req/GUID-9E571BE8-921A-4DAB-879A-8E72CDFF3358.html">https://docs.tibco.com/pub/sf_statsvcs/10.10.0/doc/html/TIB_sf_statsvcs_sys-req/GUID-9E571BE8-921A-4DAB-879A-8E72CDFF3358.html</a> for TIBCO Spotfire® Statistical Services system requirements</li> <li>• TIBCO Spotfire® Statistical Services 11.4.x server, see <a href="https://docs.tibco.com/pub/sf_statsvcs/11.4.0/doc/html/TIB_sf_statsvcs_11.4.0_sys-req/GUID-9E571BE8-921A-4DAB-879A-8E72CDFF3358.html">https://docs.tibco.com/pub/sf_statsvcs/11.4.0/doc/html/TIB_sf_statsvcs_11.4.0_sys-req/GUID-9E571BE8-921A-4DAB-879A-8E72CDFF3358.html</a> for TIBCO Spotfire® Statistical Services system requirements</li> <li>• TIBCO Spotfire® Statistical Services 12.0.x server, see <a href="https://docs.tibco.com/pub/sf_statsvcs/12.0.0/doc/html/TIB_sf_statsvcs_12.0.0_installation/installation-homepage.html">https://docs.tibco.com/pub/sf_statsvcs/12.0.0/doc/html/TIB_sf_statsvcs_12.0.0_installation/installation-homepage.html</a> for TIBCO Spotfire® Statistical Services system requirements</li> </ul> |
| TERR™ Version                        | <ul style="list-style-type: none"> <li>• TIBCO® Enterprise Runtime for R 6.0, see <a href="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">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</a> for TIBCO Spotfire® Server system requirements</li> </ul>   |