Spotfire® Enterprise Runtime for R Service Installation and Administration

Software Release 1.17.7

Contents

The Spotfire® Enterprise Runtime for R - Server Edition

Spotfire® Enterprise Runtime for R - Server Edition (a/k/a TERR service) (Formerly TIBCO Enterprise Runtime for R - Server Edition) extends access to the Spotfire Enterprise Runtime for R (a/k/a TERR™) engine from the Spotfire® Analyst (desktop product) to Spotfire® Business Author users.

TERR™ - Server Edition (a/k/a TERR™ service) is provided to Spotfire® Server administrators as a set of Spotfire SPK packages, in one SDN distribution file. Spotfire Server administrators can install and configure the TERR service on a node manager available to Spotfire Server to provide robust predictive and computational detail to users accessing Spotfire visualizations through the Spotfire® Business Author or Spotfire® Consumer.

The ability to develop predictive analytics, data functions, or TERR™ expressions is still available only to users of Spotfire Analyst.

The help for TERR service is for Spotfire Server administrators who need to install and configure the TERR service, to install R packages for visualizations that use them, and to review logs and troubleshoot the TERR service.

System Requirements

Your deployment must meet certain requirements to run the TERR service.

See the system requirements for this version of TERR service.

TERR, TERR Service, and Spotfire Statistics Services

You can run TERR data functions in Spotfire Analyst by using the local TERR engine, by using the TERR service, or by using the TERR engine provided with Spotfire® Statistics Services .

The engine Spotfire Analyst users specify depends on the statistics language used, and whether the analysis is shared with others through Spotfire Business Author and Consumer.

Make sure you use the same release of TERR for analyses across services.

TERR

Spotfire Analyst authors can create analyses that use TERR data functions or TERR custom expressions to run locally (on Windows only). They can use the built-in predictive analytics in Spotfire Analyst. They can use the TERR engine that is provided in the Spotfire Analyst authors must have the correct Spotfire license to use the TERR engine.)

To specify the local TERR engine:

1. In Spotfire Analyst, from the menu, click **Tools** > **Options**.
2. In the Options dialog box, from the left list, select **Data Functions**.

By default, the option **Use locally-installed Spotfire Enterprise Runtime for R engine** is selected.

TERR Service

The TERR service is installed on a node for your Spotfire Server. The TERR service is a licensed component of Spotfire Statistics Services. You do not need to install Spotfire Statistics Services to use the TERR service. The TERR service provides similar functionality to the Spotfire Statistics Services service for deploying TERR data functions, expression functions, or other advanced analytics; however, it is a completely separate architecture, designed to integrate with Spotfire similar to the other existing Spotfire services.

Spotfire Analyst authors can share analyses that they created using TERR data functions, TERR predictive analytics, or TERR custom expressions with users of Spotfire Business Author and Consumer. The TERR service distribution must be installed on a node for the Spotfire Server. By default, when such an analysis is shared, the TERR service provides the functionality for the users.

Prior to the release of Spotfire Statistics Services version 10.0, the Spotfire Statistics Services service was used to share analyses created using TERR data functions, expression functions, or other advanced analytics. Starting with Spotfire Statistics Services version 10.0, Spotfire recommends using TERR service instead, which is provided to Spotfire Statistics Services customers as a component on the Spotfire Download site.

Data functions written to use open-source R require Spotfire Statistics Services. They do not work with the TERR service.

Spotfire Statistics Services

If Spotfire Analyst authors are using a statistical language other than TERR (that is, SAS or MATLAB), then you must have installed and configured Spotfire Statistics Services to work with that language in your Spotfire Server environment.

To specify the URL for the Spotfire Statistics Services installation:

1. In Spotfire Analyst, from the menu, click **Tools** > **Options**
2. In the Options dialog, from the left list, select **Data Functions**.
3. In the Data Functions panel, select **Custom URL**, and in the text box, type the URL for your Spotfire Statistics Services.

If the data functions or custom expressions have been created using TERR, and if you have the TERR service installed and deployed on a node for Spotfire Server, then Spotfire always uses the TERR service, even if the analyst supplies a custom URL, and you have Spotfire Statistics Services configured to use TERR deployed in your environment. Spotfire disregards the custom URL for all analyses that use the TERR engine. This behavior provides both backwards compatibility with existing Spotfire Statistics Services + TERR installations, and makes migration to TERR service easier.

Limiting exposure of your deployment

Version 1.17.4 of the TERR service is installed on a Spotfire Server node running under Linux or Windows. The Linux installation provides the option of running the TERR service in a containerization platform.

When you install TERR service and run the TERR engine, you can take steps to protect the server deployment, to minimize the risk of unauthorized access, and to minimize the possibility of malicious acts.

Statistical engines such as TERR provide functions to access data and packages on the internet. Additionally, they have functions that access the host computer system, such as those for executing system commands, and those for reading and writing files. By their very design, these languages can expose computer systems to risk from bad actors, unless the deployer takes steps to secure the environments in which they run. We strongly recommend reviewing and implementing the practices described here.

TERR service version 1.17.4 installed on a Spotfire Server node running under Windows does not have a containerized installation available.

Restricting user access

- Run the TERR service using an account that limits network access to only required external data sources and services. (Note that taking this step can limit availability to data and package updates.)
- Always run the node manager containing TERR service as a non-root user. (That is, not as root or under an Administrative account.)
- If you are running a system where other servers have access to computers running TERR service, disable passwordless access between the server and other servers.

Configuring for tighter engine control

- Preserve the default settings for using TERR service in restricted mode with the property `terr.restricted.execution.mode: TRUE`. Note that this property is set to `TRUE` by default. See Safeguarding your environment on page 21 for more information.
- If your deployment is on a Linux server, then the default configuration for TERR service is to use containers (the property `use.engine.containers: TRUE`). Running TERR service with containers enabled prevents the engines from having access to the host system. See Containerized Service on page 6 for more information.

  Docker is available under separate software license terms and is not part of the Spotfire Server or TERR service. As such, Docker is not within the scope of your license for Spotfire Server or TERR service. Docker is not supported, maintained, or warranted in any way by Cloud Software Group, Inc. Download and use of Docker is solely at your own discretion and subject to license terms applicable to Docker.

Containerized Service

When you install the TERR service on a Linux computer, by default, it is configured to use a Docker® container. A containerized TERR service is available only for Linux installations.

To use a containerized TERR service on a Linux system, download and install Docker. (Currently, using containerized TERR service is supported only on a Linux system.)

- If you have not yet installed the TERR service, install Docker first, and then install TERR service.
- If you have already installed TERR service before installing Docker, then stop TERR service, install Docker, set the configuration to use Docker, and then restart TERR service.

The version of Docker you use depends on your Linux system. See the system requirements for the recommended version. See www.docker.com for more information about Docker.

The primary benefit of installing and using TERR service in a container is that it operates the service in a "sandbox", so the TERR engine does not have access to the host file system. (For more information about script and data function trust, see the *Spotfire® Analyst User's Guide*). Running TERR service in a container results in negligible performance impact.

Containerization of the engines does not, by default, limit access to the network. If you system supports untrusted or public users creating data functions, consider additional firewall configuration on the host system to limit container exposure to the network or internet to only necessary sites and servers. Consult your OS or Docker documentation for further guidance.

The only container framework with which we developed and tested TERR service is Docker. We do not provide Docker with the base installation; however, you must have Docker installed for TERR service to work properly out of the box. The service downloads and builds a default Docker image based on almalinux:8.9 from Docker Hub. While you cannot modify the image we provide, you can build and use a different Docker image if you have different configuration requirements. This section contains a few examples of specifying different Docker images. Alternatively, check Docker Hub for an image that might work for you.

Docker is available under separate software license terms and is not part of the Spotfire Server or TERR service. As such, Docker is not within the scope of your license for Spotfire Server or TERR service. Docker is not supported, maintained, or warranted in any way by Cloud Software Group, Inc. Download and use of Docker is solely at your own discretion and subject to license terms applicable to Docker.

You can export and change the configuration options to build and install a customized image. See Configuring the Service on page 16 for more information. The properties to change in the configuration are the following.

- `docker.image.name`
- `startup.hook.script`

Important After you install Docker, you must create the **docker** group and then add the **spotfire** user to the docker group. For more information about setting up this group, see https://docs.docker.com/engine/install/linux-postinstall/.

After you install and configure both the container and TERR service, you can start the service. When the service starts, containers are created as needed.

Optionally, you can set the configuration option `use.engine.containers` to `FALSE` to disable the container option.

Configuring a custom Docker image on a node with internet access

If you have access to the internet, then you can build a Docker image for your TERR service, referencing it by its name and tag.

Perform this task from the command line on the computer where your Spotfire Server is installed. Some steps are performed on the computer where the node manager is installed. (This can be a different computer.)

You can create the custom configuration before installing the TERR service.

Prerequisites

- You must have Docker installed on the computer running the node manager. If you install and start the service before you install Docker, then exceptions are written to the log.
- You must have a Linux computer where the node manager installed. (Your node manager and Spotfire Server can be on different computers).
- If you are using the script to build the base Docker image, you must have a connection to the internet. (A connection to the internet is not required if you are using a locally-available Docker image.)

Procedure

1. If you have already installed the service from the Spotfire Server Nodes & Services administration page, and if it is running, then stop the service.
2. On the node manager computer, create a new directory called `docker`. Inside that directory, create your `Dockerfile`.

   Important Remember that for any script you write, the line endings must be appropriate for the operating system where the service runs. Many text editors can perform end-of-line (EOL) conversion.

```
############################################
# A sample Dockerfile for a startup script. #
#############################################
FROM almalinux:8.9
```

```
# install openjdk
RUN yum update -y && yum install -y java-17-openjdk && yum clean all
# set JAVA_HOME variable
ENV JAVA_HOME=/usr/lib/jvm/jre
```

For more information, see https://docs.docker.com/engine/reference/builder/.

3. On the computer running the node manager, build the image with the name and tag.

   The name and tag are comprised as `<name:version>`, as follows.

   ```
   docker build -t terrsrv:258 .
   ```

   For more information, see https://docs.docker.com/engine/reference/commandline/build/.

4. On the computer running Spotfire Server, export the `custom.properties`.
5. Edit the settings in the file `custom.properties`, specifying the name and tag of your custom image.

   ```
   use.engine.containers: TRUE
   docker.image.name: <name:version>
   ```

6. On the computer running Spotfire Server, import the `custom.properties`.
7. From the Spotfire Server Nodes & Services administration page, install the service, specifying the configuration to use, and then start the service.

   If you have already installed the service, then in the node manager, select the service and click **Edit**. From the **Configuration** drop-down list, select the new configuration.

If problems occur, troubleshoot by examining the `Dockerfile` that the service writes. After the service runs, this `Dockerfile` is available at the root service directory on the computer running the node manager. For example, `/opt/nodemanager/<version>/nm/services/<language>-service-linux-<version_#_ID>/dockerfile/Dockerfile`.

Configuring a custom Docker image on a node with no internet access

If your node manager does not have external access to the internet, then you can create a Docker image on an internet-enabled computer, and then transfer it to your node manager.

Perform the first three steps of this task from the command line on a computer with internet access. Perform the rest of the task from the command line on the computer where your node manager is installed.

Prerequisites

- If you are using the script to build the base Docker image, you must have a connection to the internet. (A connection to the internet is not required if you are using a locally-available Docker image.)
- You must have a Linux computer where the node manager installed. (Your node manager and Spotfire Server can be on different computers).
- You must have Docker installed on the computer running the node manager. If you install and start the service before you install Docker, then exceptions are written to the log.

Custom docker images for the service must contain the following.

- The Java 17 Runtime.
- The JAVA_HOME environmental variable, correctly defined.

  ```
  ENV JAVA_HOME=</correct/path/to/java>
  ```

Procedure

1. On a computer with internet access, create the `Dockerfile`.

   Important Remember that for any script you write, the line endings must be appropriate for the operating system where the service runs. Many text editors can perform end-of-line (EOL) conversion.

   ```
   ############################################
   # A sample Dockerfile for a startup script. #
   #############################################
   # install openjdk
   RUN yum update -y && yum install -y java-17-openjdk && yum clean all
   # set JAVA_HOME variable
   ENV JAVA_HOME=/usr/lib/jvm/jre
   ```

   For more information, see https://docs.docker.com/engine/reference/builder/.

2. Build the image specifying the name and tag.

   Use the command `docker build -t` *`<name:version>`*, as follows.

   ```
   docker build -t terrsrv:258 .
   ```

   For more information, see https://docs.docker.com/engine/reference/commandline/build/.

3. Save the image to a `.tar` file.

   Use the command `docker save -o` *`<name-version>`*`.tar` *`<name:version>`*, as follows.

   ```
   docker save -o terrsrv-258.tar terrsrv:258
   ```

   For more information, see https://docs.docker.com/engine/reference/commandline/save/.

4. If you have already installed the service from the Spotfire Server Nodes & Services administration page, and if it is running, then stop the service.

5. Transfer the `.tar` file to the target computer (where the node manager is running).

6. Load the `.tar` file into the node manager.

   Use the command `docker load -i` *`<name-version>`*`.tar`, as follows.

   ```
   $ sudo docker load -i terrsrv-258.tar
   f2419d350464: Loading layer [==================================================>]
    329.5MB/329.5MB
   Loaded image: terrsrv:258
   $ docker images
   REPOSITORY                TAG                   IMAGE ID            CREATED             SIZE
   terrsrv                   258                   9941b68e7f65        17 hours ago        517MB
   ```

   For more information, see https://docs.docker.com/engine/reference/commandline/load/.

7. On the computer running Spotfire Server, export the `custom.properties`.

8. Edit the settings in the file `custom.properties`, specifying the name and tag of your custom image.

   ```
   use.engine.containers: TRUE
   docker.image.name: <name:version>
   ```

9. On the computer running Spotfire Server, import the `custom.properties`.

10. From the Spotfire Server Nodes & Services administration page, install the service, specifying the configuration to use, and then start the service.

    If you have already installed the service, then in the node manager, select the service and click **Edit**. From the **Configuration** drop-down list, select the new configuration.

See Installing the Service on a Node Manager for a Spotfire Server on page 14 and Configuring the Service on page 16 for more information.

Configuring a custom startup script to build a custom Docker image

You can provide a startup script that is installed and configured on your Spotfire Server build a custom Docker image for your TERR service.

Perform this task from the command line on the computer where your Spotfire Server is installed, and on the computer where your node manager is installed. For more information about the startup script, see Startup script on page 23.

Prerequisites

- You must have Docker installed on the computer running the node manager. If you install and start the service before you install Docker, then exceptions are written to the log.
- You must have a Linux computer where the node manager installed. (Your node manager and Spotfire Server can be on different computers).
- If you are using the script to build the base Docker image, you must have a connection to the internet. (A connection to the internet is not required if you are using a locally-available Docker image.)

Custom docker images for the TERR service must contain the following.

- The Java 17 Runtime.
- The JAVA_HOME environmental variable, correctly defined.

```
ENV JAVA_HOME=<correct/path/to/java>
```

Procedure

1. If you have already installed the service from the Spotfire Server Nodes & Services administration page, and if it is running, then stop the service.
2. On the computer running Spotfire Server, export the `custom.properties`.
3. On computer running Spotfire Server, create a file called Dockerfile, and then save it to your custom configuration directory.

   Important Remember that for any script you write, the line endings must be appropriate for the operating system where the service runs. Many text editors can perform end-of-line (EOL) conversion.

```
#############################################
# A sample Dockerfile for a startup script. #
##############################################
# install openjdk
RUN yum update -y && yum install -y java-17-openjdk && yum clean all
# set JAVA_HOME variable
ENV JAVA_HOME=/usr/lib/jvm/jre
```

4. On the computer running Spotfire Server, create a custom script to build the `Dockerfile`, and then save it to your custom configuration directory.

   (By default, `<server-installation-dir>/tomcat/spotfire-bin/config/root/conf`.)

   The following example file is named `customScript.sh`.

```
#!/bin/bash

    # Define the image name and tag
IMAGE_NAME="terrsrv:customScript"
    # Custom configuration files are at relative path conf/FILE
DOCKERFILE_NAME="conf/Dockerfile"
    # Command to check if image exists
COMMAND="docker inspect ${IMAGE_NAME}"
```

```
    # Run the command then check the status code
$COMMAND
RESULT=$?
if [ $RESULT -ne 0 ]; then
 # Image did not exist
 echo ${IMAGE_NAME} does not exist. Building now...
 COMMAND="docker build -f ${DOCKERFILE_NAME} -t ${IMAGE_NAME} ."
 echo ${COMMAND}
 echo "Building the custom docker image ${IMAGE_NAME} for the terr-
service"
 $COMMAND
 echo "Completed building ${IMAGE_NAME}"
else
 # Image exists already
 echo The requested image ${IMAGE_NAME} already exists.
fi
```

5. Edit the relevant properties in the file `custom.properties`, specifying using the custom script.

   ```
   use.engine.containers: TRUE
   docker.image.name: terrsrv:customScript
   startup.hook.script: conf/customScript.sh
   ```

6. On the computer running Spotfire Server, import the `custom.properties`.
7. From the Spotfire Server Nodes & Services administration page, install the service, specifying the configuration to use, and then start the service.

   If you have already installed the service, then in the node manager, select the service and click **Edit**. From the **Configuration** drop-down list, select the new configuration.

If problems occur, troubleshoot by examining the `Dockerfile` that the service writes. After the service runs, this `Dockerfile` is available at the root service directory on the computer running the node manager. For example, `/opt/nodemanager/<`*`version`*`>/nm/services/<`*`language`*`>-service-linux-<`*`version_#_ID`*`>/dockerfile/Dockerfile`.

Pulling a custom Docker image from an authenticated repository

You can create a custom start script to configure TERR service to log in to a remote authenticated repository and pull a custom Docker image.

This option is available if you want to specify a base image for the docker container, but it is in a repository that requires authentication to access. To set the appropriate authentication credentials, you can execute a Docker login command when you start the service, but before starting the Docker container, as part of a startup hook script.

This task demonstrates accessing a Docker image stored in the AWS Elastic container Registry, which is an authenticated repository.

Prerequisites

- You must have a Linux computer where the node manager installed. (Your node manager and Spotfire Server can be on different computers).

Custom docker images for the service must contain the following.

- The Java 17 Runtime.
- The JAVA_HOME environmental variable, correctly defined.

  ```
  ENV JAVA_HOME=</correct/path/to/java>
  ```

Procedure

1. If you have already installed the service from the Spotfire Server Admin UI Nodes and Services page, and if it is running, then stop the service.

2. Install the AWS command-line interface (CLI) tool on the computer running the node manager.

   See https://docs.aws.amazon.com/cli/latest/userguide/awscli-install-bundle.html for more information.

   a) Run the command `aws configure`, and then connect to your account using your AWS Access Key and AWS Secret Access Key.

   b) Verify that the user running the TERR service can run the `aws` process.

3. Determine your `docker.image.name` property.

   a) In your AWS account, navigate to **Amazon ECR > Respositories**.

      The docker image name is listed after `Repository URI`, and the tag is listed after `Image Tags`.

      ```
      Repository URI 123456.dkr.ecr.us-west-2.amazonaws.com/terr/terrsrv-sample
      Image Tags: latest
      ```

   The `docker.image.name` property is a concatenation of those two values.

   ```
   docker.image.name: 123456.dkr.ecr.us-west-2.amazonaws.com/terr/terrsrv-sample:latest
   ```

4. On the computer running Spotfire Server, export the `custom.properties`.

5. On the computer running Spotfire Server, create a custom script and save it to your custom configuration directory `<`*`server installation dir`*`>/tomcat/spotfire-bin/config/root/conf/`.

   The script uses the AWS `get-login` command to fetch the `docker login` command. See the following links for more information.

   - https://docs.aws.amazon.com/cli/latest/reference/ecr/get-login.html
   - https://docs.docker.com/engine/reference/commandline/login/

   In the script, use the absolute path to the `aws` command (`usr/local/bin/aws`).

   We named this sample script `awsScript.sh`.

   If saved to a custom configuration, it resides at the relative path `conf/awsScript.sh`

   Important Remember that for any script you write, the line endings must be appropriate for the operating system where the service runs. Many text editors can perform end-of-line (EOL) conversion.

   ```
   #!/bin/bash
       # Request a login from AWS
       # The command will return a 'docker login' string
   DOCKER_LOGIN=`/usr/local/bin/aws ecr get-login --no-include-email --region us-west-2`
   echo Retrieved the command ${DOCKER_LOGIN}
       # Execute that 'docker login'
   ${DOCKER_LOGIN}
   echo docker login authentication completed.
   ```

6. From the command line, manually test your script at this stage to ensure that everything works correctly.

7. Edit the relevant properties in the file `custom.properties` with the appropriate values.

   ```
   docker.image.name: 123456.dkr.ecr.us-west-2.amazonaws.com/terr/terrsrv-sample:latest
   use.engine.containers: TRUE
   startup.hook.script: conf/awsScript.sh
   ```

8. From the command line, manually test the script again to make sure that it works correctly.

9. On the computer running Spotfire Server, import the `custom.properties`.

10. From the Spotfire Server Nodes & Services administration page, install the service, specifying the configuration to use, and then start the service.

    If you have already installed the service, then in the node manager, select the service and click **Edit**. From the **Configuration** drop-down list, select the new configuration.

What to do next

If problems occur, troubleshoot by examining the `Dockerfile` that the service writes. After the service runs, this `Dockerfile` is available at the root service directory on the computer running the node manager. For example, `/opt/nodemanager/<`*`version`*`>/nm/services/<`*`language`*`>-service-linux-<`*`version_#_ID`*`>/dockerfile/Dockerfile`.

Installing the Service on a Node Manager for a Spotfire Server

After installing and authorizing a node manager, you can install the TERR service.

Prerequisites

- You have installed and authorized a node manager. See the topics Node manager installation and Trusting a node in *Spotfire® Server Installation and Administration*.
- Spotfire Server and the node manager are up and running.
- You have deployed the operating-system-specific SDN for TERR service into the deployment area of your Spotfire Server. For information about deploying the SDN, see the topic Adding software packages to a deployment area in *Spotfire® Server Installation and Administration* .

  Important
  - You can install TERR service on a node manager running on a computer with an operating system (OS) that is different from that of your Spotfire Server.
  - All services belonging to the same Spotfire deployment area must run on node managers with the same OS.
- Optional: You have created and imported a custom configuration for TERR service.

Procedure

1. Log in to Spotfire Server, select your deployment area if needed, and click **Nodes & Services**.
2. Under **Node managers**, select the node to which you want to add TERR service.
   A running service displays a green circle with a check mark next to the selected node manager.
   In the **Services** area, the names of the services are displayed in the lower-right pane of the window.
3. Click **Create new service**.
4. Make your selections in the Create new service dialog:
   a) Under **Deployment area**, select the area where you deployed TERR service.

      Administrators often create a Test deployment area to use as a staging server.

   b) Under **Capability**, select **TERR**.
   c) Under **Configuration**, select the service configuration to apply to the service.

      In most cases, this is the default configuration, unless you have created a custom configuration. See the *Spotfire® Server Installation and Administration Help* for more information on creating a custom configuration.

   d) Under **Service name**, provide a display name for the service.
   e) Under **Add instances**, specify the number of instances.
   f) Under **Instances name**, provide a name for the instances.
   g) Under **Number of instances**, leave the option set to `1`.
      TERR service can have only one instance per node. If you set it to a value other than `1`, the service does not work as expected.

      One TERR service instance can serve multiple users simultaneously. See the Custom configuration properties for more information.
   h) Under **Port**, you can change the default as needed.

5. Click **Create service**.

   To view the progress of the installation, click the **Activity** tab.

Result

The service is installed and starts.

Additionally, three helper R packages are installed. See Helper packages included in the service on page 15.

If you experience errors, click **View logs** for more information.

What to do next

For information on the remaining setup tasks, see the topic Post-installation steps in the *Spotfire® Server Installation and Administration* help.

Helper packages included in the service

The following packages are included with the service to provide functionality for working with Spotfire.

Package	Description
SpotfireData	Contains functions for reading and writing sbdf files.
SpotfireUtils	Contains functions that users can call directly from a script.
SpotfireSPK	Contains a function for building and distributing Spotfire (SPK) packages. See The Spotfire Package (SPK) on page 26

For more information about the functions in these packages, load the package and read its help file.

Example

1. On the computer running the node manager, open the R console.
2. From the prompt, set the package path.

```
.libPaths(c(.libPaths(), "<NM_HOME>/services/<path-to-spotfire-service-for-r>/library/
<ver>"))
```

3. Load the package.

```
library(SpotfireData)
```

4. Load the help.

```
help(SpotfireData)
```

Configuring the Service

You can customize certain behaviors for the TERR service by exporting the service properties, editing the file, importing service properties, and then applying the new configuration.

Perform this task on the computer where you have installed Spotfire Server.

For general information about configuring services for Spotfire Server, see the Spotfire® Server Installation and Administration.

Prerequisites

- You must have completed the steps outlined in Installing the Service on a Node Manager for a Spotfire Server on page 14.
- You must have administrative read-write privileges to save the changed configuration file.

Procedure

1. Open a command line as administrator and change the directory to the location of the command-line config tool (on Windows, `config.bat`; on Linux, `config.sh`).

   The default location is `<`*`server installation dir`*`>/tomcat/spotfire-bin`.

2. On the command line, issue the following command.

   ```
   config export-service-config --capability=TERR --deployment-area=<your deployment area
    name>
   ```

   If you already have a configuration name from previously editing the configuration, and you want to change that configuration, provide the configuration name using the `--config-name=<`*`configuration name`*`>` option.

   The file named `custom.properties` is exported and written to the directory `<`*`server installation dir`*`>/tomcat/spotfire-bin/config/root/conf`.

3. When prompted, provide the password for the config tool.

4. Using a text editor, open and edit the file `<`*`server installation dir`*`>/tomcat/spotfire-bin/config/root/conf/custom.properties`.

   The text file contains comments to provide you with information about each property. Alternatively see the individual reference topics for the properties for more information.

5. Save the changes, and then close the text editor.

6. Optional: Copy any additional files to add to the configuration into the directory `<`*`server installation dir`*`>/tomcat/spotfire-bin/config/root/conf/`.

   For example, you can add a configuration script. (Configuration scripts must be specified in the custom property `startup.hook.script`. See Startup script on page 23 for more information.). For a Linux OS deployment, you can add a Dockerfile.

7. From a command line, return to the directory for the command-line config tool.

   The default location is `<`*`server installation dir`*`>/tomcat/spotfire-bin`.

8. On the command line, issue the following command.

   ```
   config import-service-config --config-name=<new-config-name>
   ```

   The `config-name` you specify identifies this configuration, so provide a name that is meaningful for the change. For example, if you create a configuration with a specific debugging level, you might name the configuration Debugging.

   You cannot overwrite the default configuration. You must provide a configuration name when you import the custom configuration.

   See the reference topic for `import-service-config` in the *Spotfire® Server and Environment Installation and Administration* help for information about additional options.

9. Open a web browser and log in to the administration console for Spotfire Server.
10. Click **Nodes & Services**.
11. Under Network, select **Node managers**, and then select the TERR service.
12. Click **Edit**.
    The Edit service dialog is displayed.
13. In the **Configuration** drop-down list box, select the configuration name to apply, and then click **Save**.

Result

The service is stopped, and then Spotfire Server restarts the service and applies the new configuration. The TERR service begins recording information to the Service Logs. For more information, see Service Logs on page 40.

To change the new configuration, export it again, specifying its name. If you do not specify the name, the default configuration is exported.

Custom configuration properties

You can fine tune the behavior of the TERR service by setting custom configuration properties.

Allowed engines

You can specify the number of TERR engines that can run concurrently, and the number of TERR engines that are allocated in the TERR service queue.

Configuration property	Default setting	Description
`engine.session.max`	*<one less than the number of logical processors available on the node>*	The maximum number of TERR engine sessions that are allowed to run concurrently in the TERR service. Each user running data functions in a Spotfire analysis uses its own session. The default is one less than the number of logical processors on the host.
`engine.queue.size`	*<one quarter of the number of logical processors on the host, constrained to a minimum of 1 and a maximum of 10. >*	The number of TERR engines preallocated and available for new sessions in the TERR service queue. The service always starts enough engines to keep the queue at the requested level. The total number of engines that can run at the same time is the sum of `engine.session.max` + `engine.queue.size`. This number can be set manually to a value higher than 10.

For more information on how engine resources can be managed, see Service resource management scenarios on page 38.

Compressed job contents and results

You can compress large data sets sent to TERR. You can also compress returned results for TERR data functions that are then sent to TERR service.

Configuration property	Default setting	Description
`jetty.gzip.compression-level`	`4`	Set to a value from `1` to `9`, inclusive. 1 offers the fastest compression speed but at a lower ratio. `9` offers the highest compression ratio but at a lower speed.
`jetty.gzip.min-gzip-size`	`32`	The minimum size in bytes before the response is compressed.
`jetty.gzip.inflate-buffer-size`	`2048`	The size, in bytes, of the buffer to inflate a compressed request. Set to `-1` to disable compression uploads.

Disable warnings

By default, warnings from the data function (or included packages) are sent to the client when executing a data function through the service. It is possible to suppress warnings by disabling the warning alert.

Configuration property	Default setting	Description
`disable.warning.alert`	`false`	Set to `true` to disable sending warnings from the data function to the client when executing a data function.

Engine pruning

When the TERR service reaches a certain percentage of capacity of usage, then the TERR service begins pruning TERR engines to free service resources.

Configuration property	Default setting	Description
`engine.prune`	`10`	The time, in seconds, that a TERR engine can be idle before the TERR service prunes it.
`dynamic.prune.threshold`	`60`	The TERR service capacity at which idle pruning is engaged, as a percentage value. By default, when the TERR service reaches 60% capacity of usage, then it begins the idle-pruning process as specified by `engine.prune`. • Set to `0` to always prune when a TERR engine is idle. • Set to `100` to never prune when a TERR engine is idle.

For more information, see Service resource management scenarios on page 38.

Engine timeout

You can specify the length of time a TERR engine runs to complete a task before failing with a timeout error. You can also specify the length of time for a TERR session to exist.

Configuration property	Default setting	Description
`engine.execution.timeout`	`600`	The length of time, in seconds, that the TERR service allows the TERR engine to execute a request before stopping the execution with a timeout error.
`engine.session.maxtime`	`1800`	The length of time, in seconds, that the TERR service allows the TERR engine session to exist before killing it. To disable session pruning, set this value to `-1`.

File size upload limit

When planning for uploading files for the TERR service, you can set the file size limit for uploading using the properties setting for the Spring Boot framework. If you change this setting, consider how the file size might affect the speed at which files can be uploaded.

Configuration property	Default setting	Description
`spring.servlet.multipart.max-file-size`	`100 MB`	The total file size for upload cannot exceed the value for this setting.
`spring.servlet.multipart.max-request-size`	`100 MB`	The total request size for a multipart file upload cannot exceed the value for this setting.

Logging level

By default, the logging level is set for the TERR service to provide informational progress.

In the `custom.properties` file, you can set the logging level through the property `loggingLevel`. The TERR service uses Log4J2-defined logging levels.

Level	Description
`ALL`	All levels are reported.
`TRACE`	Reports a finer-grained level of events than the `DEBUG` level.
`DEBUG`	Reports a fine-grained level of events. This setting is most useful when you are debugging problems with the service.
`INFO`	The default. Reports informational messages that highlight the progress of the TERR service, but at coarse-grained level.
`WARN`	Reports potentially harmful situations.
`ERROR`	Reports errors. These errors might still allow the TERR service to continue running.
`FATAL`	Reports only very severe errors that cause the TERR service to stop.
`OFF`	Turns off logging.

Manage Java options

You can set certain Java command-line options for the TERR service for managing such settings as the Java heap size.

Configuration property	Default setting	Description
`disable.java.core.dump`	`TRUE`	By default, when the JVM stops responding, it does not write full core dumps to the `temp` directory. Set this value to `FALSE` to enable full core dumps. Setting this value to `FALSE` can potentially cause the core dump to fill all available disk space. Use with caution.

`javaOptions`	*none*	In systems with a lot of memory, administrators might want to limit the initial or maximum heap size that TERR service can use.
		
		In the following example, the custom property sets the Java initial heap size to 1GB.
		
		`javaOptions:-Xms1g`
		
		In the following example, the custom property sets the Java initial heap size to 2GB and the maximum heap size to 4GB.
		
		`javaOptions:-Xms2g,-Xmx4g`
		
		The `javaOptions` setting controls Java memory only, and does not control memory allocated in the TERR runtime. (Memory allocated by Java is typically very small, which makes it possible to set a very low value for the initial heap size, such as `Xms64M`.)
		
		To include multiple java options, delimit the options with a comma and no space. The following examples demonstrate setting an empty property, setting one property, setting two properties, and setting three properties.
		
		`javaOptions:`
		`javaOptions:-Xms2g`
		`javaOptions:-Xms2g,-Xmx4g`
		`javaOptions:-Dfoo="foobar",-Xms2g,-Xmx4g`
		
		The `javaOptions` property cannot contain spaces. For example, `-Dfoo="foo bar"` is not a valid property setting.
		
		For other Java command line options, see the Java documentation.

Package library location

You can set the location of packages that TERR can use in the TERR service configuration settings.

Configuration property	Default setting	Description
`packagePath`	none	The absolute path to the shared package library location. When specifying the path, you must use a forward slash, regardless of operating system. • If you install packages into your Spotfire deployment using the SPK process, then this setting is not required. • If you install packages onto the server using the TERR function `install.packages(<package-name>)`, then set this path to the package installation location.

Example

```
packagePath: /opt/terr/library
packagePath: C:/Progra~/spotfire/terr61/library
```

For more information, see Package Management for the Service on page 25.

Safeguarding your environment

This custom property setting helps minimize the risk of malicious acts in your environment.

Configuration property	Default setting	Description
`disable.spotfire.trust.checks`	`FALSE`	By default, TERR service checks whether a data function has come from a trusted source. Set to `TRUE` to not check for the data function trust status of any data function run on TERR service. ⚠ Setting this value to `TRUE` results in all Spotfire data functions executing unrestricted. We strongly recommend that you ensure that your service is fully secured, that engine containers are enabled, and that network access from the containers is limited (using a firewall) to only necessary servers and ports. For more information about script and data function trust, see the *Spotfire® Analyst User's Guide* and *Spotfire® Administration Manager User's Guide*.

Configuration property	Default setting	Description
`terr.restricted.execution.mode`	`TRUE`	This setting causes each expression sent to the TERR engine, running in TERR service, to be evaluated by the function `>>name>>` as to whether it is a restricted operation. Restricted execution mode allows executing arbitrary scripts without worrying that the script could do malicious things, such as deleting files or uploading confidential data to a server over the internet. `evalREX` is fairly conservative, preventing many operations, while still allowing some useful exceptions. The following shows a non-exhaustive list of operations disallowed in restricted execution mode. If such an evaluation is attempted, TERR generates an error, such as `"Error: restricted call to Native[tempfile]"`, and execution of the expression is terminated. • Calling the restricted TERR function `evalREX` itself. • Performing any I/O to the file system or the internet. • Loading new packages, except for the libraries included with TERR (stats and so on). • Spawning new operating system processes (that is, calling `system`). • Calling `.Call`, used to call Rapi code in CRAN packages. • Calling `.C` or `.Fortran`. • Calling into Java using the terrJava package (which allows executing arbitrary Java methods). • Calling any functions in the parallel package (which uses terrJava). • Accessing any function environments in the stack above the call to `evalREX` using either `sys.frame` or `parent.frame`. This prevents malicious code from installing functions or expressions that could be executed after leaving restricted execution mode. • Changing the variable lookup path by setting `parent.env` of an environment, or reading or setting the environment of a closure. • Defining S4 classes and methods using `setClass` or `setMethod`. If you need to use any part of the above functionality as part of your TERR service deployment, then you can set `terr.restricted.execution.mode=FALSE`. If you change this setting, be sure to review your deployment for additional steps you can take to ensure against malicious acts. See Limiting exposure of your deployment for more information. For more information about `terrUtils::evalREX`, see its help file in the *Spotfire® Enterprise Runtime for R Language Reference*.

Startup script

You can specify a script to run before a container or TERR engine is started.

Configuration property	Default setting	Description
`startup.hook.script`	none	The path and name of a startup hook script that runs before any container or TERR engine is started. This value can be empty (for no script) or a relative path from TERR service working directory. Place the startup script in the directory with the `custom.properties` file (by default `<`*`server-installation-dir`*`>/tomcat/spotfire-bin/config/root/conf/`). To specify the path, you must use forward slash (/) regardless of the host operating system. TERR service can run either a `.bat` or a `.sh` file format, depending on host operating system. On a Linux® system, the script must have appropriate permissions before TERR service executes it. **Important** Remember that for any script you write, the line endings must be appropriate for the operating system where the service runs. Many text editors can perform end-of-line (EOL) conversion. You can use the startup script to set environment variables, create directories, download files, or prepare the file system settings in other ways before the service starts. (For example, you can perform Docker commands in the script for a Linux deployment area, or you can run another script. See Containerized Service on page 6 for more information.)

Example

- Relative path for a Linux deployment area: `conf/`*`mystartupscript`*`.sh`
- Relative path for a Windows deployment area: `conf/`*`mystartupscript`*`.bat`

Engine ports

TERR engines running under the TERR service require open ports to communicate. The first available port, and range to the last available port are determined by these two settings.

The defaults specify a range between 61001 and 62000.

Configuration property	Default setting	Description
`engine.port.min`	`61001`	The first specified available port set for a TERR engine.
`engine.port.range`	`1000`	This value, added to the value specified in `engine.port.min`, indicates the range of the ports available for the TERR engines.

JMX monitoring

You can use an installation of Java Management Extensions (JMX) and the Remote Method Invocation (RMI) connector to monitor the TERR service.

Remove the comment marker and set the properties to connect to JMX using RMI in the custom properties file. To use JMX monitoring, you must provide valid settings for all five of these properties.

Important Because JMX monitoring requires connecting to the specific IP address of the node, you must create a custom configuration for each node to monitor.

Configuration property	Default setting	Description
`jmx.rmi.username`	None	Set this value to the JMX user name.
`jmx.rmi.password`	None	Set this value to the JMX password.
`jmx.rmi.host`	None	Set this value to the IP address of the host computer (that is, the computer where the node manager and the TERR service are installed.)
`jmx.rmi.port`	`1099`	Set this value to an available port for the RMI connection with JMX.
`jmx.active`	None	Set this value to `TRUE` to activate JMX.

For more information, see Monitoring the Service using JMX on page 41.

Containerized configuration

TERR service provides custom properties that are specific to the Linux operating system.

Configuration property	Default setting	Description
`use.engine.containers`	`TRUE`	Runs the TERR engine inside a container when this value is set to `TRUE` (the default).
`ram.limit`	`1000`	The amount of RAM and SWAP memory to which the TERR engine containers are constrained, in megabytes.
`docker.image.name`	`almalinux:8.9`	When you use containers, the TERR service builds a custom image based on a starting image. This property is used by the Dockerfile as the `FROM` line. See https://docs.docker.com/engine/reference/builder/#from for more information. TERR service default is almalinux:8.9.

For more information, see Containerized Service on page 6.

Package Management for the Service

If a data function author in your organization creates a Spotfire analysis that uses packages from a public or private repository, then the administrator must make sure that the packages are installed on Spotfire Server.

Important Packages must be installed in the same type of environment as where they are executed. For example, if you built the almalinux:8.9 Docker image for your containerized TERR service, make sure the packages are installed on a node manager running the same operating system as the containerized engines. If this is not possible, then see the workaround options in Troubleshooting the Service on page 43.

TERR programmers in your organization can develop their own packages or take advantage of some of the thousands of compatible packages developed by other TERR programmers, and then share the analyses that use those functions with Spotfire users in your organization.

Data function authors must be in the Script Author group to create and save data functions that use TERR.

The largest and most commonly-used, curated repository for R packages is the Comprehensive R Archive Network (CRAN). Most packages that are not installed with the TERR service are downloaded from this repository.

Testing results for CRAN package compatibility are available for Windows and Linux.

Important By default, the TERR engine runs in "Restricted Execution" mode, where attempting to execute certain operations defined as restricted (such as installing packages or performing any I/O operation) generates an error. This option is meant to protect your system from potentially malicious code running in the global environment. To enable your data function developers to upload and use packages, you must allow the TERR service to run in unrestricted mode. For more information about restricted mode, see the TERR language reference for `evalREX`. For more information on setting trust options for data functions and scripts in Spotfire, see the topic *Script and Data Function Trust* in the Spotfire Analyst Help.

The service provides the following ways to add packages.

Installing packages directly on the node manager computer, using the TERR engine installed there.
This way makes the packages accessible to a TERR service engine running on a node when the engine session is created. This method is the most efficient, because calling `install.packages()` directly on the server ensures that all dependent packages are also installed. If you use this method, you must specify the package path in the custom configuration file. See Configuring the Service for more information.

Building an SPK that contains custom-designed or CRAN packages that are compatible with the supported TERR engine.
The SPK is uploaded to the server. Packages are stored as part of the deployment and are accessible to the TERR service engine running on a node when the engine session is created. This way of installing packages requires that you create and manually maintain a Debian Control File (DCF) that lists all packages to be installed, including dependent packages. Because the DCF controls SPK versioning, it provides better package governance than directly uploading to the server. See The Spotfire Package (SPK) on page 26 for detailed instructions for creating an SPK and managing its packages. See Installing R Packages Manually on page 35 for more information.

Important If you create an SPK on a computer other than the computer where the node manager is installed, then be sure to use a computer that is running same operating system as the computer where the packages are going to be deployed. For example, If you are creating an SPK on Linux, be sure your Linux OS type and version are the same as that of the computer where the node manager is installed.

Configuring the `packagePath` in `custom.properties` to point to a mounted drive or directory that contains packages that are compatible with the TERR engine.

This way of package management is the most controlled, where only packages in the specified repository are used. Data function authors must also configure their systems to write their data functions to use these packages.

Regardless of which way your organization uses to manage packages, the administrator who installs the TERR packages must do the following.

- Coordinate with the author of a Spotfire data function to ensure that all TERR packages that are used in the analysis are installed on the host computer running the nodes on which the TERR service is running.
- Ensure that all required and dependent packages associated with the needed packages are also installed. (The author of the data function can provide the administrator with this information.)

The SPK package management tools handle installing all required and dependent packages needed by a specified package.

Important If you install packages from the TERR console, or using TERR provided with Spotfire Analyst, you usually install them using a script such as `install.packages(c("`*`packagename`*`", "`*`packagename`*`"))`. However, if you try to install packages using such a script from the TERR service then the packages are not available to the entire service. To ensure that the packages you need are available to the entire service, you need to install them on the host computer running the TERR service.

Find help

Spotfire includes many avenues to help with packages, whether they are TERR language packages to use with TERR service or Spotfire packages (SPKs).

Task	Help resource
Building packages using Spotfire Enterprise Runtime for R.	*Spotfire® Enterprise Runtime for R Technical Documentation* at https://docs.tibco.com/products/spotfire-enterprise-runtime-for-r
Deploying a package using the Spotfire package mechanism.	*Spotfire® Server Installation and Administration* at https://docs.tibco.com/products/spotfire-server
Learning about Spotfire Statistics Services architecture and server management, versus the TERR service management.	*Spotfire® Statistics Services Installation and Administration* at https://docs.tibco.com/products/spotfire-statistics-services

The Spotfire Package (SPK)

A Spotfire SPK is usually created and tested by developers to package and deploy third-party extensions to the Spotfire Server, which can then be distributed to the Spotfire Server node for use by another service (and in some cases, distributed to Spotfire clients).

Even though they are both called “packages”, the R package and the Spotfire package (SPK) are different.

- The R package (usually downloaded from a repository, such as CRAN) contains specialized R functions.
- The SPK is a means to deploy extensions to the Spotfire Server, which either distributes its contents to Spotfire Analyst users, or installs for a service, such as TERR service, to use from the Spotfire Server node.

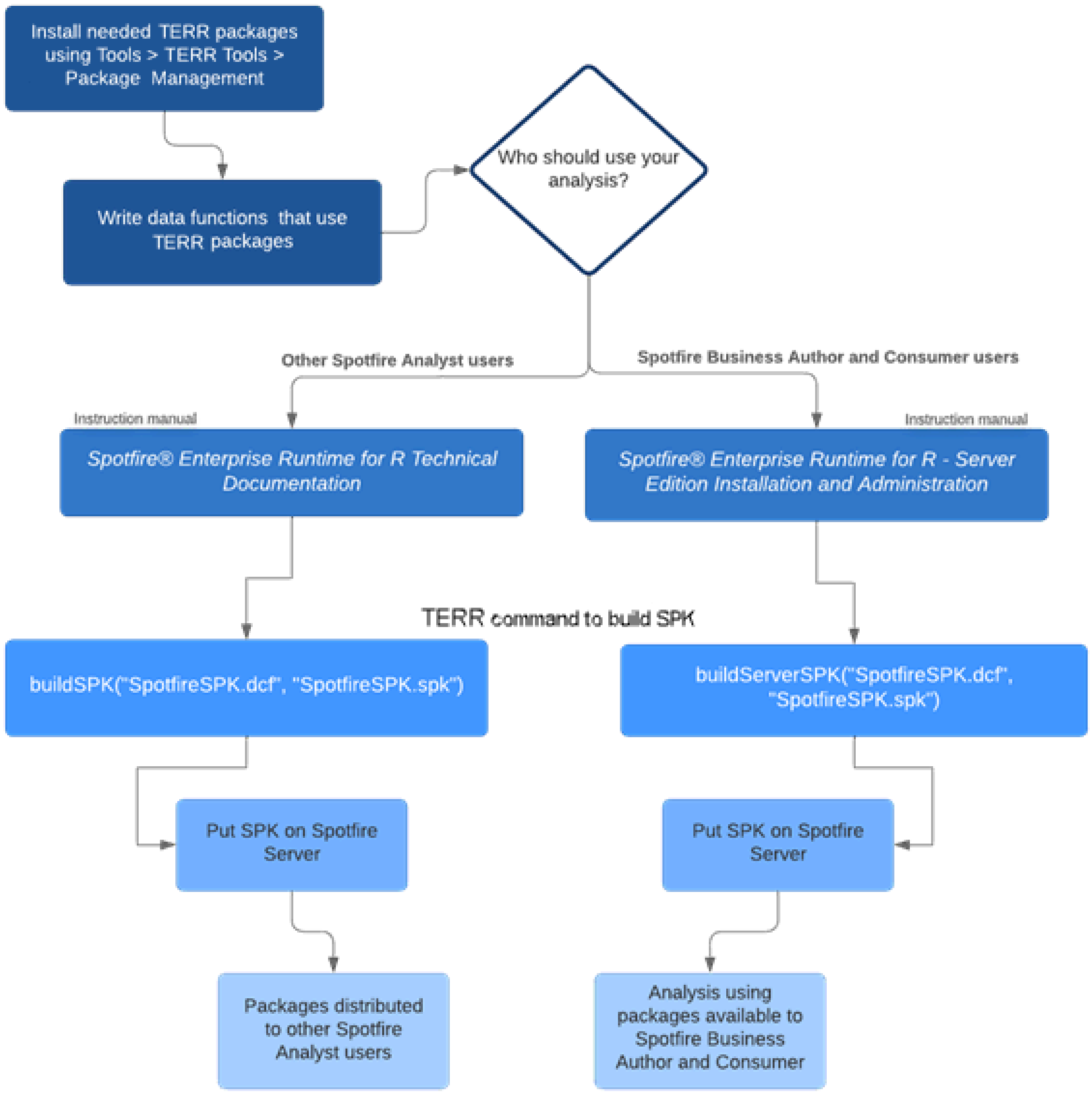
To make it easy to create SPK files containing TERR functions, use the package SpotfireSPK, which has the following two functions:

- `buildSPK`, which creates an SPK containing packages suitable for distribution to other Spotfire Analyst clients to create analyses using data functions.

  This use case applies only to TERR and not TERR service, so it is not covered in the TERR service documentation.

- `buildServerSPK`, which creates an SPK containing packages suitable for distribution to a Spotfire Server for use by the TERR service.

This documentation does not cover Spotfire SPKs in general, only those packages built using the SpotfireSPK package that contain R packages. For more general information about extending Spotfire using the SPK, see the Spotfire Developer documentation available at docs.tibco.com.


Install needed TERR packages using Tools > TERR Tools > Package Management
Write data functions that use TERR packages
Who should use your analysis?
Other Spotfire Analyst users
Spotfire Business Author and Consumer users
Instruction manual
Spotfire® Enterprise Runtime for R Technical Documentation
Instruction manual
Spotfire® Enterprise Runtime for R - Server Edition Installation and Administration
TERR command to build SPK
buildSPK("SpotfireSPK.dcf", "SpotfireSPK.spk")
buildServerSPK("SpotfireSPK.dcf", "SpotfireSPK.spk")
Put SPK on Spotfire Server
Put SPK on Spotfire Server
Packages distributed to other Spotfire Analyst users
Analysis using packages available to Spotfire Business Author and Consumer

Obtaining the SpotfireSPK toolset

Using the SpotfireSPK package, you can generate a Spotfire SPK containing your TERR packages, and then automatically distribute them to Spotfire Analyst installations in your organization.

You must have installed Spotfire Analyst. You must have a license to use TERR in your Spotfire Analyst installation.

Procedure

1. From the Spotfire Analyst menu, click **Tools** > **TERR Tools**.
2. On the **Engine** tab, click **Launch TERR Console**.
   The TERR console in the Spotfire installation is displayed.
3. At the console prompt, type `library("SpotfireSPK")`.
   The package is loaded in the TERR session, and you are now ready to build the SPK to contain the TERR-compatible binary packages.
4. Review the Help for the package functions, `buildSPK` and `buildServerSPK`, by typing the following in at the command-line in the TERR console:
   `?buildSPK` or `?buildServerSPK`
   - `buildSPK` builds an SPK that is distributed to other Spotfire Analyst users.
   - `buildServerSPK` builds an SPK that is installed on the Spotfire Server for the TERR Service to use.

Spotfire SPK versioning

Typically, you create the SPK (`.spk`) file that contains the packages, and then you install it on a service running on a Spotfire Server node. You might need to change or update the packages that you distribute, which requires updating the `.spk` file.

You can create or change a Spotfire `.spk` file using TERR or from a text editor. The TERR functions `buildSPK` and `buildServerSPK` create the Spotfire `.spk` file using the versioning rule details for the following tasks.

The SPK property `BuiltVersion` is NOT the same as the package version. That information is stored in the package `DESCRIPTION` file. `BuiltVersion` is always specified as four components (`N.n.n.n`).

You can learn more about the TERR functions `buildSPK` and `buildServerSPK` by reading their help files. See Obtaining the SpotfireSPK toolset for more information.

You can learn more about the TERR function `buildServerSPK` by reading its help file.

As of TERR version 5.1, the SPK you create by calling `buildSPK` or `buildServerSPK` also includes Imports and Depends packages.

Task	Example	Notes and Results

Create a new DCF for a new SPK containing packages to put on the Spotfire Server.	From the TERR console, install the required packages, and then create the SPK.
	`#load the library containing`
	`#the tools`
	`library(SpotfireSPK)`
	
	`#install the packages to go`
	`# into the SPK`
	`install.packages(c("plyr","zoo"))`
	
	`#create the DCF containing`
	`# the package names`
	`writeLines("Packages: plyr,zoo", "`
	`ServerSPK.dcf")`
	
	`#create the SPK to upload to the`
	`# Spotfire Server`
	`buildServerSPK("ServerSPK.dcf", "`
	`ServerSPK.spk")`
		A DCF with the name `ServerSPK.dcf` is created containing the package names listed in the `text` argument of `writeLines`, along with their required and dependent packages. The specified packages are included in the SPK.
		The SPK `BuiltVersion` number is listed in the DCF as follows.
		`"BuiltVersion: 1.0.0.0"`
		You can see the `BuiltVersion` by opening the DCF in a text editor, or by calling `readLines`:
		`readLines("ServerSPK.`
		`dcf")`

Add packages to an existing SPK to put on Spotfire Server.	Using a text editor, open the existing DCF, and add names to the list of the (installed) packages to put in the SPK. Do not edit any other part of the DCF.
	`Packages: plyr,zoo,caret,forecast`
	From the TERR console, create the SPK.
	`#create the SPK to upload to the`
	`# Spotfire Server`
	`buildServerSPK("ServerSPK.dcf", "`
	`ServerSPK.spk",`
	`spkName = "Official CRAN Packages")`
		The DCF with the name `ServerSPK.dcf` is overwritten, and the packages you manually added to the DCF are added to the SPK.
		The SPK `BuiltVersion` is changed to the next minor version number and is listed in the DCF. For example:
		`"BuiltVersion: 1.1.0.0"`

Remove a package from the existing SPK (and subsequently from Spotfire Server).	Using a text editor, open the existing DCF, and remove the unwanted package names from the list of packages. Do not edit any other part of the DCF.
	`Packages: plyr`
	From the TERR console, create the SPK.
	`#create the SPK to upload to the`
	`# Spotfire Server`
	`buildServerSPK("ServerSPK.dcf", "`
	`ServerSPK.spk",`
	`spkName = "Official CRAN Packages")`
		The DCF with the name `ServerSPK.dcf` is overwritten, and only the packages remaining in the DCF are included in the SPK.
		The SPK `BuiltVersion` is changed to the next major version number and listed in the DCF. For example:
		`"BuiltVersion: 2.0.0.0"`

Task	Example	Notes and Results

Assign a specific `BuiltVersion` number to an SPK.	From the TERR console, install the required packages, and then create the SPK.
	`#load the library containing`
	`# the tools`
	`library(SpotfireSPK)`
	
	`#install the required packages`
	`install.packages(c("plyr","zoo"))`
	
	`#create the DCF containing`
	`# the packages`
	`writeLines("Packages: plyr,zoo", "`
	`ServerSPK.dcf")`
	
	`#create the SPK to upload to the`
	`# Spotfire Server,`
	`#passing in the version number.`
	`#This argument is a character string`
	` or a`
	`#numeric_version object containing`
	` four components`
	`buildServerSPK("ServerSPK.dcf", "`
	`ServerSPK.spk", version="1.2.3.4")`
		The DCF with the name `ServerSPK.dcf` is created (or overwritten if it already existed) , and the packages specified in the DCF are included in the SPK.
		The SPK `BuiltVersion` is set to the value passed in for the argument version. For example:
		`"BuiltVersion: 1.2.3.4"`

Generate a new DCF with the same name.	From the TERR console, install the required packages, and then create the SPK.
	`#load the library containing`
	`# the tools`
	`library(SpotfireSPK)`
	
	`#install the packages to go into`
	`# the SPK`
	`install.packages(c("forecast","caret"`
	`))`
	
	`#create the DCF containing the`
	`# names of the installed`
	`# packages`
	`writeLines("Packages: forecast,caret"`
	`, "ServerSPK.dcf")`
	
	`#create the SPK to upload to the`
	`# Spotfire Server`
	`buildServerSPK("ServerSPK.dcf", "`
	`ServerSPK.spk")`
		Important If you use this method to overwrite an existing DCF, the version number is still set to `1.0.0.0`; therefore, Spotfire Server does not register the package as a new one, so it does not distribute the new packages to the users.
		The DCF with the name `ServerSPK.dcf` is overwritten, and the package names listed in the `text` argument of `writeLines`, along with their required and dependent packages, are included. The packages specified are included in the SPK.
		The SPK `BuiltVersion` number is listed in the DCF as follows:
		`"BuiltVersion: 1.0.0.0"`

Creating an SPK to put a CRAN Package on Spotfire Server

By running a few lines of TERR code, you can create a Spotfire SPK that you can deploy to Spotfire Server for use by the Spotfire web clients.

You can create an SPK containing packages, you can add a package to an existing SPK, or you can remove a package from an SPK. This topic demonstrates creating an SPK containing two packages by downloading the packages, and then creating the Debian Control File (DCF) that specifies the package contents.

Perform this task from the TERR console.

You must use the same version of TERR that is running in the TERR service on the node manager. This version of TERR must be a supported version, as listed in the System Requirements.

On Windows, start the TERR console available from the **Tools** menu in Spotfire Analyst.

Prerequisites

- Review the SPK information in this Help.
- You must have access to the TERR console.
- You must have access to the Spotfire Server deployment area.
- You must have access to an internet connection that allows you to download packages from CRAN.
- You must build a package for the operating system on which your Spotfire Server node manager is installed.

Procedure

1. From the TERR console, install the packages you want to include in the Spotfire SPK.

   This example shows installing two packages. The function that builds the SPK includes all Includes and Depends packages in the SPK.

   ```
   install.packages(c("dplyr","zoo"))
   ```

   If you are running Windows, you can install the packages you want in Spotfire Analyst from the **TERR Tools** Package Management dialog box.

2. In the TERR console, generate a list file.

   In this example, we create the Debian Control File (DCF), which is a list file that contains the two sample packages.

   ```
   writeLines("Packages: dplyr,zoo", "ServerSPK.dcf")
   ```

   The file `ServerSPK.dcf` contains one line: `"Packages: plyr,zoo"`. (Make a note of the file location.)

3. Load the library that contains the functions to create the SPK for Spotfire Server.

   ```
   library(SpotfireSPK)
   ```

4. In the TERR console, call the function that creates the SPK, and pass in the name of the DCF you created, and then add the packages to an SPK named `ServerSPK.spk`.

   ```
   buildServerSPK("ServerSPK.dcf", "ServerSPK.spk")
   ```

   The Spotfire package file is created, and the file `ServerSPK.dcf` now contains the information similar to the following:

   ```
   Packages: dplyr,zoo
   Built: TERR service <ver>; 2024-02-14 15:35:37 PST
   BuiltFile: ServerSPK.spk
   BuiltName: TERR™ - Server Edition (a/k/a TERR™ service) Packages for Spotfire
           Server Windows c36b8e61
   BuiltId: c36b8e61-8967-40ac-9d7a-d8656b03eb91
   BuiltVersion: 1.0.0.0
   BuiltPackages: R6 (>=2.4.1),assertthat (>=0.2.1),cli (>=2.1.0),crayon
           (>=1.3.4),digest (>=0.6.25),dplyr (>=1.0.2),ellipsis
           (>=0.3.1),fansi (>=0.4.1),generics (>=0.0.2),glue
           (>=1.4.2),lattice (>=0.20-41),lifecycle (>=0.2.0),magrittr
           (>=1.5),pillar (>=1.4.6),pkgconfig (>=2.0.3),purrr
           (>=0.3.4),rlang (>=0.4.8),tibble (>=3.0.4),tidyselect
           (>=1.1.0),utf8 (>=1.1.4),vctrs (>=0.3.4),zoo (>=1.8-8)
   ```

   The `BuiltVersion` is always `1.0.0.0` when you create the DCF using TERR, and then to create an SPK, unless you specify a different version in the function call. See Spotfire SPK versioning for more information.

   The SPK is ready to be placed in the Spotfire Server deployment area. Make a note of its location.

5. Open a browser window and log in to Spotfire Server as the administrator.
6. Click **Deployments & Packages**.
7. Under **Software Packages**, click **Add Packages**.
   The Add packages dialog box is displayed.
8. Browse to the location of the new SPK file, and then select it and click **Open**.
9. Click **Upload**.
10. Click **Save area** to save the deployment area.

Result

The packages are added to the Spotfire Server node manager where the TERR service is deployed. They are ready to use in a visualization published for Spotfire web client users.

If you already installed the TERR service to your node manager, you must upgrade the service for the newly-installed packages to be deployed.

Adding CRAN Packages to an existing SPK to put on Spotfire Server

By editing the DCF and then running a few lines of TERR code, you can add packages to a Spotfire SPK that you can deploy to Spotfire Server for use by the TERR service and the Spotfire web clients.

You can create an SPK containing packages, you can add a package to an existing SPK, or you can remove a package from an SPK. This topic demonstrates adding two packages to an existing SPK by first downloading the packages, and then adding their names to the DCF that specifies the package contents, and finally uploading the resulting SPK to the Spotfire Server.

Perform this task from the TERR console. Always use the version of TERR that is listed as supported in the System Requirements.

On Windows, start the TERR console available from the **Tools** menu in Spotfire Analyst.

Prerequisites

- You must have access to the TERR console.
- You must have access to the original DCF that contains the packages in the original SPK.

  The DCF is written to the default location for `writeLines()`, unless you specify another location.
- You must have access to the Spotfire Server deployment area.
- You must have access to an internet connection that allows you to download packages from CRAN.
- You must build a package for the operating system on which your Spotfire Server node manager is installed.

Procedure

1. From the TERR console, install the packages you want to add.

   This example shows adding two packages. The function that builds the SPK includes all Includes and Depends packages in the SPK.

   ```
   #example
   install.packages(c("caret", "forecast"))
   ```

   If you are running Windows, you can install the packages you want in Spotfire Analyst, from the **Tools** Package Management dialog box.

2. Open the DCF containing the original packages in a text editor.
3. Find the line containing the packages.
   `"Packages: plyr,zoo"`.
4. Add the new package names to the list, and then save the DCF.

   Do not change any other lines in the DCF.

   `"Packages: plyr,zoo,caret,forecast"`.
5. In the TERR console, call the function that creates the SPK, and pass in the name of the DCF you edited and the file name of the original SPK. Additionally, explicitly provide the name of the package to display in the **Deployments & Packages** list on Spotfire Server.

   The name of the SPK to display in the **Deployments & Packages** list is needed to differentiate the manually-edited file from the built file. You can provide the default, or you can provide a different string.

   ```
   buildServerSPK("ServerSPK.dcf", "ServerSPK.spk", spkName="Approved CRAN Packages")
   ```

   The Spotfire package file is created, and the file `ServerSPK.dcf` now contains the new packages, as well as all Includes and Depends packages, and the BuiltVersion is incremented by a minor version number.

   ```
   "Packages: plyr,zoo,caret,forecast"
   "Built: TERR service <ver>; 2024-02-14 16:04:01 PST
   BuiltFile: ServerSPK.spk
   BuiltName: Approved CRAN Packages
   BuiltId: c36b8e61-8967-40ac-9d7a-d8656b03eb91
   BuiltVersion: 2.1.0.0
   BuiltPackages: KernSmooth (>=2.23-17),MASS (>=7.3-53),Matrix
           (>=1.2-18),ModelMetrics (>=1.2.2.2),R6 (>=2.4.1),RColorBrewer
           (>=1.1-2),Rcpp (>=1.0.5),SQUAREM (>=2020.4),TTR
           (>=0.24.2),assertthat (>=0.2.1),caret (>=6.0-86),class
           (>=7.3-17),cli (>=2.1.0),codetools (>=0.2-16),colorspace
           (>=1.4-1),crayon (>=1.3.4),curl (>=4.3),data.table
           (>=1.13.0),digest (>=0.6.25),dplyr (>=1.0.2),ellipsis
           (>=0.3.1),fansi (>=0.4.1),farver (>=2.0.3),foreach
           (>=1.5.0),forecast (>=8.13),fracdiff (>=1.5-1),generics
           (>=0.0.2),ggplot2 (>=3.3.2),glue (>=1.4.2),gower
           (>=0.2.2),gtable (>=0.3.0),ipred (>=0.9-9),isoband
           (>=0.2.2),iterators (>=1.0.12),labeling (>=0.3),lattice
           (>=0.20-41),lava (>=1.6.8),lifecycle (>=0.2.0),lmtest
           (>=0.9-38),lubridate (>=1.7.9),magrittr (>=1.5),mgcv
           (>=1.8-33),munsell (>=0.5.0),nlme (>=3.1-149),nnet
           (>=7.3-14),numDeriv (>=2016.8-1.1),pROC (>=1.16.2),pillar
           (>=1.4.6),pkgconfig (>=2.0.3),plyr (>=1.8.6),prodlim
           (>=2019.11.13),purrr (>=0.3.4),quadprog (>=1.5-8),quantmod
           (>=0.4.17),recipes (>=0.1.13),reshape2 (>=1.4.4),rlang
           (>=0.4.8),rpart (>=4.1-15),scales (>=1.1.1),stringi
           (>=1.5.3),stringr (>=1.4.0),survival (>=3.2-7),tibble
           (>=3.0.4),tidyr (>=1.1.2),tidyselect (>=1.1.0),timeDate
           (>=3043.102),tseries (>=0.10-47),urca (>=1.3-0),utf8
           (>=1.1.4),vctrs (>=0.3.4),viridisLite (>=0.3.0),withr
           (>=2.3.0),xts (>=0.12.1),zoo (>=1.8-8)
   ```

   The SPK is ready to be placed in the Spotfire Server deployment area.
6. Open a browser window and log in to Spotfire Server as the administrator.
7. Click **Deployments & Packages**.
8. Under **Software Packages**, click **Add Packages**.
   The Add packages dialog box is displayed.
9. Browse to the location of the new SPK file, and then select it and click **Open**.
10. Click **Upload**.
11. Click **Save area** to save the deployment area.

Result

The packages are added to the Spotfire Server node manager where the TERR service is deployed. They are ready for use in analyses to be opened with the Spotfire web client.

If you already installed the TERR service to your node manager, you must upgrade the service for the newly-installed packages to be deployed.

Removing CRAN Packages from an existing SPK to put on Spotfire Server

By editing the DCF and then running a few lines of TERR code, you can remove packages from a Spotfire SPK that you can deploy to Spotfire Server for use by the TERR service and Spotfire web client users.

You can create an SPK containing packages, you can add a package to an existing SPK, or you can remove a package from an SPK. This topic demonstrates removing packages from an existing SPK, and then uploading the resulting SPK to the Spotfire Server.

Perform this task from the TERR console. Always use the version of TERR that is listed as supported in the System Requirements.

On Windows, start the TERR console available from the **Tools** menu in Spotfire Analyst

Prerequisites

- Review the SPK information in this Help.
- You must have access to the TERR console.
- You must have access to the Spotfire Server deployment area.
- You must have access to an internet connection that allows you to download packages from CRAN.
- You must build a package for the operating system on which your Spotfire Server node manager is installed.

Procedure

1. Open the DCF containing the original packages in a text editor.
2. Find the line containing the package names.
   `"Packages: plyr,zoo,caret,forecast"`.
3. Remove the package names from the list, and then save the DCF.

   Do not change any other lines in the DCF.

   `"Packages: forecast"`.
4. In the TERR console, call the function that creates the SPK, and pass in the name of the DCF you edited, and the file name of the original SPK.

   Additionally, explicitly provide the name of the package to display in the **Deployments & Packages** list on Spotfire Server. The name of the SPK to display in the Deployments and Packages list is needed to differentiate the manually-edited file from the built file.

   ```
   buildServerSPK("ServerSPK.dcf", "ServerSPK.spk", spkName="Approved CRAN Packages")
   ```

   The Spotfire package file is created, and the file `ServerSPK.dcf` now contains only the packages you retained in the file, as well as all Includes and Depends packages, and the BuiltVersion is incremented by a major version number.

   ```
   "Packages: forecast"
   "TERR service <ver>; 2024-02-14 16:00:14 PST
   BuiltFile: ServerSPK.spk
   BuiltName: Approved CRAN Packages
   ```

```
BuiltId: c36b8e61-8967-40ac-9d7a-d8656b03eb91
BuiltVersion: 2.0.0.0
BuiltPackages: MASS (>=7.3-53),Matrix (>=1.2-18),R6
        (>=2.4.1),RColorBrewer (>=1.1-2),Rcpp (>=1.0.5),TTR
        (>=0.24.2),assertthat (>=0.2.1),cli (>=2.1.0),colorspace
        (>=1.4-1),crayon (>=1.3.4),curl (>=4.3),digest
        (>=0.6.25),ellipsis (>=0.3.1),fansi (>=0.4.1),farver
        (>=2.0.3),forecast (>=8.13),fracdiff (>=1.5-1),ggplot2
        (>=3.3.2),glue (>=1.4.2),gtable (>=0.3.0),isoband
        (>=0.2.2),labeling (>=0.3),lattice (>=0.20-41),lifecycle
        (>=0.2.0),lmtest (>=0.9-38),magrittr (>=1.5),mgcv
        (>=1.8-33),munsell (>=0.5.0),nlme (>=3.1-149),nnet
        (>=7.3-14),pillar (>=1.4.6),pkgconfig (>=2.0.3),quadprog
        (>=1.5-8),quantmod (>=0.4.17),rlang (>=0.4.8),scales
        (>=1.1.1),tibble (>=3.0.4),timeDate (>=3043.102),tseries
        (>=0.10-47),urca (>=1.3-0),utf8 (>=1.1.4),vctrs
        (>=0.3.4),viridisLite (>=0.3.0),withr (>=2.3.0),xts
        (>=0.12.1),zoo (>=1.8-8)
```

The SPK is ready to be placed in the Spotfire Server deployment area.

5. Open a browser window and log in to Spotfire Server as the administrator.
6. Click **Deployments & Packages**.
7. Under **Software Packages**, click **Add Packages**.
   The Add packages dialog box is displayed.
8. Browse to the location of the new SPK file, and then select it and click **Open**.
9. Click **Upload**.
10. Click **Save area** to save the deployment area.

Result

The packages are added to the Spotfire Server node manager where the TERR service is deployed. They are ready for use in analyses to be opened with the Spotfire web client.

If you already installed the TERR service to your node manager, you must upgrade the service for the newly-installed packages to be deployed.

Installing R Packages Manually

If you have write permission to the computer hosting TERR service, then you can install packages to a specific location manually, and then point to the package location in the custom configuration.

Perform this task on the computer hosting the TERR service (in the directory where the TERR engine is installed), and then on the computer where Spotfire Server is installed.

This method of installing packages is useful for packages that have a larger number of dependent packages. Dependent packages are installed by default, rather than manually, using this method.

Any time you install additional packages or update existing packages, be sure to install them in the directory you specified for your `packagePath`. You can have only one package path for TERR service installation.

Prerequisites

You must have administrative privileges to create a directory where packages are installed. You must have administrative privileges and the tools password to update the `custom.properties` file.

Procedure

1. On the host computer, create a directory to hold the packages.

   You need to perform this step just the first time you install packages.

   Example: `/opt/r/library`

2. From a command prompt, launch TERR console.

   You can find the TERR engine in the `/nodemanager/<version_#>/nm/services/<TERR Service name_guid#> /terr/bin` directory.

3. Install the packages to the directory you created, and where the TERR service can find them.
   In this example, the survival package and its dependent packages are installed in the directory `/opt/terr/library`.

   ```
   install.packages("survival", lib="/opt/terr/library")
   ```

   The package and its dependent packages are installed.

4. Close the TERR console.
   At the TERR command prompt, run the command `q()`.
   The TERR console closes.

5. Update TERR service configuration to specify the package path.

   You need to export, edit, and reimport the `custom.properties` file only the first time to set the package path.

   Remember that when you change the `custom.properties`, you must edit the service configuration with a new name, and then reimport it to have it take effect. See Manually editing the service confguration files for more information.

   a) Follow steps 1-3 in Configuring the Service on page 16 to export the service configuration file `custom.properties`.

   b) In the exported `custom.properties` file, locate the entry for `packagePath`.

   c) Provide the path that you specified for the installed packages.

      The configuration setting `packagePath` requires forward slashes (/) regardless of operating system.

   d) Complete the steps to rename, save, update, and import the changed service configuration file, as described in Configuring the Service on page 16.

Creating a Spotfire Analysis to Display Installed Packages

You can create a data function for a Spotfire analysis that displays a table of all packages installed and available to the TERR engine running in your TERR service.

Perform this task in an instance of Spotfire Analyst, and then save the resulting analysis to the Spotfire library on the server.

Procedure

1. Open any analysis.

2. From the menu, click **Tools** > **Register Data Functions**.
   The Register Data Functions dialog box is displayed.

3. Provide a name, such as `Installed Packages`.

4. From the **Type** drop-down list box, select **R script - Open Source R**.

5. Clear the **Allow caching** check box

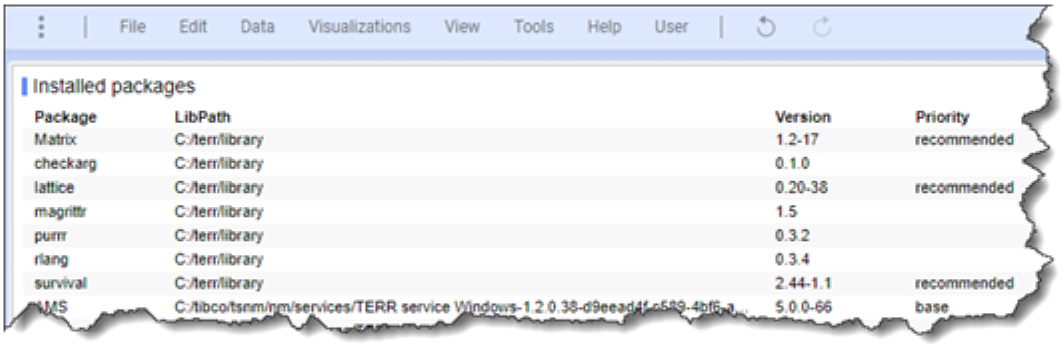
6. In the **Script** text box, type the following function.
   `packages <- installed.packages()`

7. Click the **Output Parameters** tab, and then click **Add**.
   The Output Parameter dialog box is displayed.

8. For **Result parameter name**, type `packages`.
9. Provide a **Display name**, such as `Installed Packages`.
10. From the **Type** drop-down list box, select **Table**.
11. Run the function, and in the resulting Edit Parameters dialog box, select the **Refresh function automatically** check box, and optionally, specify whether you want the function to always run in a separate engine session. (For more information about the option **Always run in separate session**, see "Configuring Data Function Parameters" in the *Spotfire® User's Guide*.)
12. In the **Run location** drop down list box, select **Force Server**.
13. Click the **Output** tab, and for the **Output handler**, select **Data table**.
14. Click **OK** and then save the data function to the library on the server.

Result

When you open the analysis from the library, the resulting table reports all of the packages currently available to the TERR service on this server.



Installed packages

Package	LibPath	Version	Priority
Matrix	C:/terr/library	1.2-17	recommended
checkarg	C:/terr/library	0.1.0	
lattice	C:/terr/library	0.20-38	recommended
magrittr	C:/terr/library	1.5	
purrr	C:/terr/library	0.3.2	
rlang	C:/terr/library	0.3.4	
survival	C:/terr/library	2.44-1.1	recommended
MASS	C:/tibco/tsnm/nm/services/TERR service Windows-1.2.0.38-d9eead4f-c589-4bf6-a...	5.0.0-66	base

Service resource management scenarios

You can use a combination of the TERR service custom properties, including the pruning properties `engine.prune` and `dynamic.prune.threshold`, to ensure the best usage of the TERR engines that the TERR service allocates.

The custom properties `engine.session.max` and `engine.queue.size` determine the number of engines that are available, and the number of engines allowed in the queue, respectively. These values are determined by the number of logical processors available on the node where TERR service is running. Additionally, you might want to set properties that control how long a TERR engine in a session can remain idle, how long to run an execution before timing out, or the percentage of engines that can run in a session before pruning is triggered.

The following two configuration examples, and their associated scenarios, demonstrate the resource management for different combinations of custom properties. These non-exhaustive usage scenarios are provided only to give two of many configurations for engine pruning and engine idle timeout. Your needs can vary, depending on your job sizes, the number of users, and the number of available logical processors.

Configuration A

Assume the following configuration values, where three engines are created and waiting in the queue for jobs.

```
#Configuration A
engine.execution.timeout: 60
engine.session.maxtime: 120
# by default, these are set to number of logical processors, minus 1, on the system
engine.session.max: 3
engine.queue.size: 3
# the idle timeout
engine.prune: 10
# The service capacity at which idle pruning is engaged, as a percentage value.
#   0 = always idle prune.
# 100 = never idle prune.
dynamic.prune.threshold: 100
```

The following three scenarios show how this configuration affects the jobs that users submit.

Scenario	Result
1A: A single user submits a job that runs for more than 60 seconds	Because this job runs for longer than the value set for `engine.execution.timeout`, the execution is halted and the engine is destroyed. Results for this long job are not returned. When the user submits another job, a new engine is provided from the queue.
2A: A single user submits a job that runs for 5 seconds	The job completes and returns results, and the engine persists. The capacity of the service is equal to the number of engines in use divided by the maximum sessions. In this case, the capacity is 1/3, or 33%, which is below the `dynamic.prune.threshold` value of `100`. The user can access this same engine for up to `engine.session.maxtime` (in this case `120` seconds).
3A: Four users submit jobs that execute for 5 seconds each	The first three user jobs get engines, and the fourth user job must wait for an engine to become available, because the `engine.session.max` has been reached. Because `dynamic.prune.threshold` is set to `100`, which specifies never idle prune, the first three users claim their engines for the duration of the value of the `engine.session.maxtime` of `120` seconds. At this point, the engines are destroyed and new engines are made available on a first-come-first-served basis.

Configuration B

Change the configuration as follows, where only the `dynamic.prune.threshold` has been changed from `100` to `0`.

```
#Configuration B
engine.execution.timeout: 60
engine.session.maxtime: 120
# by default, these are set to number of logical processors on the system
engine.session.max: 3
engine.queue.size: 3
# the idle timeout
engine.prune: 10
# The service capacity at which idle pruning is engaged, as a percentage value.
#   0 = always idle prune.
# 100 = never idle prune.
dynamic.prune.threshold: 0
```

The same user scenarios show how this changed configuration affects the jobs that users submit.

Scenario	Result
1B: A single user submits a job that runs for more than 60 seconds	No change in behavior from Configuration A.
2B: A single user submits a job that runs for 5 seconds	The job completes and return results. The capacity is at 33 which is now higher than the `dynamic.prune.threshold` of `0`, so the engine is destroyed after the `engine.prune` (idle timeout) of `10` seconds. If the user submits a new job, a new engine is created from the queue.
3B: Four users submit jobs that execute for 5 seconds each	The first three user jobs get engines, and the fourth user job must wait for an engine to become available, because the `engine.session.max` has been reached. However, `dynamic.prune.threshold` is set to always idle prune (`0`), so after submitting a job and getting results, if a user sits idle for longer than `engine.prune` of `10` seconds, the engine is destroyed. More engines are created and made available to the fourth user.

Conclusion

In scenario 3A (where four users submit jobs, and the `dynamic.prune.threshold` is set to `100`), the fourth user might have to wait for up to 2 minutes for an available engine (the `engine.session.maxtime`), whereas in scenario 3B (where four users submit jobs, and the `dynamic.prune.threshold` is set to `0`), the fourth user could wait for just 15 seconds (job run of 5 seconds and `engine.prune` idle timeout of `10` seconds).

By default, the `dynamic.prune.threshold` is set to `60`, because this setting balances both access for a high volume of users and faster response times for a lower volume of users. The default values for `engine.execution.timeout` and `engine.session.maxtime` are set to balance security and availability. For your on-premises usage, you might find it useful to increase execution timeouts or disable idle timeouts altogether.

See Engine pruning on page 18 for more information about these custom properties. See Configuring the Service on page 16 for information about setting all custom properties.

Service Logs

After the TERR service is installed and started, it begins writing to the logs. These logs are stored in the directory `<node manager installation>/logs`.

Log name	Description
`service-<instanceid>-stdout.log`	Prints `INFO`-level information about the service startup, shut down, and any exceptions. It also prints the Jetty component to the standard log files. To set Jetty logging to `DEBUG` level, in the startup script (`start-terr-service.sh` for Linux nodes, `startTerr.bat` for Windows nodes), uncomment the command `SET JETTY_DEBUG=-Dorg.eclipse.jetty.LEVEL=DEBUG`.
`TERR-service-<instanceid>.log`	Prints configuration options that the TERR service starts with. Provides granular level of the individual engines, job execution details.

Monitoring the Service using JMX

The TERR service supports JMX monitoring integration. JMX monitoring is turned off by default.

Prerequisites

You can install and use JConsole for monitoring the TERR service using JMX. JConsole is provided as part of the Java SE Development Kit. (See Using JConsole from the Oracle Java documentation site.) Alternatively, you can install and use VisualVM to monitor the TERR service using JMX.

Important Because JMX monitoring requires connecting to the specific IP address of the node, you must create a custom configuration for each node to monitor.

Procedure

1. Stop the TERR service.
2. Export and edit the `custom.properties`, setting the following properties.

   ```
   jmx.rmi.username: username
   jmx.rmi.password: password
   jmx.rmi.host: <IP address of the Node Manager running Spotfire Service for R>
   jmx.rmi.port: 1099
   jmx.active: TRUE
   ```

   See Configuring the Service on page 16 for detailed instructions.
3. Start the TERR service.
4. Check the `INFO` logs for the connection string.

   If the setup and connection are successful, a JMX connection string is printed to logs at the `INFO` level.

   ```
   2022-06-09T21:03:11,520 | INFO  | [main] c.s.s.t.ServiceConfig: Service configured JMX Connection
    string: service:jmx:rmi://10.10.100.60:1099/jndi/rmi://10.10.100.60:1099/jmxrmi
   ```

   - If `jmx.rmi.username`, `jmx.rmi.password`, or `jmx.rmi.host` are blank, then a log message is printed indicating that the property is blank, and that the JMX connection is not created.
   - If `jmx.rmi.port` is blank or undefined, then the port value defaults to `1099`.
   - If the `jmx.rmi.host` is configured incorrectly, the connection times out and the service fails to start. An error message is printed to the admin UI and the logs.

   A successful client connection is printed to logs at the `DEBUG` level. An unsuccessful client connection attempt due to bad or missing username or password is printed at the `ERROR` level.

   If you are connecting to a remote host, the port must be opened in the firewall to allow the connection.
5. Open JConsole, and in the **remote process** field, provide the JMX connection string provided by the logs as shown.

   ```
   service:jmx:rmi://10.10.100.60:1099/jndi/rmi://10.10.100.60:1099/jmxrmi
   ```
6. Provide the user name and password that you set in `custom.properties`.

   If a message is displayed indicating `the connection could not be made using SSL. Would you like to try without SSL?` , then click **Insecure connection**.

   JConsole should now display information from the service.
7. To view metrics specific to the TERR service, click the tab **MBeans**.

8. In the left panel, expand the group labeled **metrics**.
   Metrics are listed, including many JVM metrics. Some of the metrics specific for TERR are as follows.

```
serviceQueueCurrentSize - total number of engines currently waiting in the queue
serviceQueueEnginesDestroyed - total number of engines destroyed after successful use
serviceQueueEnginesFailed - total number of engines that failed on startup due to
 configuration, environmental, or other exceptions
serviceQueueEnginesInUse - total number of engines currently executing
serviceQueueEnginesStarted - total number of successful engines started
serviceQueueEnginesStarting - total number of engines currently initializing
serviceQueueIdealSize - the ideal queue size as defined by engine.queue.size in
 custom.properties
serviceQueueLastPortSelected - the last port chosen for engine creation
serviceUsageBytesDownloaded - total bytes downloaded through the service
serviceUsageBytesUploaded - total bytes uploaded through the service
serviceUsageCapacity - the current capacity of the service as a percentage: current
 session over maximum allowed concurrent sessions
serviceUsageJobs - total number of jobs the service has created and run
serviceUsageSessions - total number of sessions the service has created
serviceUsageMillisInUse - total time spent executing successful jobs, in milliseconds
```

Troubleshooting the Service

If you have problems with the service, review these tips.

Problems with the `Dockerfile` on a Linux node manager

After the service runs, you can view the `Dockerfile` that TERR service writes. You can find the file in the root service directory (for example, `/opt/spotfire/nodemanager/<version_#>/nm/services/terr-service-linux-<version_#_ID>/dockerfile/Dockerfile`.

Try test building the Docker image in the environment before starting TERR service.

Building an image takes time, so it can take a few minutes for the web UI to display a possible image build failure. If a build failure occurs, the retry mechanism is triggered automatically.

Errors running data functions from a custom package in a container

If you are running a data function using a custom package in Spotfire from TERR service running in a Linux container, and you encounter the error message, "`Error executing <functionname> for dynamic library <libname> from package <packagename> loaded from /opt/packages`, then your custom package probably requires a library that is not available to the generic AlmaLinux image supplied by default for your docker container. (For example, some libraries require compilation when they are installed, and they link in libraries on the host operating system that do not match the container operating system.)

To test this issue, set the custom configuration property `use.engine.containers` to `FALSE`, restart the service, and then try running the data function again. This option runs the data function in the engine that is accessible to the operating system and that can access system libraries.

If the data function runs successfully, and if you want to use containers, then you can create a custom container that uses a version of AlmaLinux compatible to your system operating system. See Configuring a custom Docker image on a node with internet access for more information.

Problems with the startup script

Check your script line endings.

Important Remember that for any script you write, the line endings must be appropriate for the operating system where the service runs. Many text editors can perform end-of-line (EOL) conversion.

TERR service, Web Player, or node becomes unresponsive on a regular basis

This problem can occur if you have installed TERR service on a node with another Spotfire service, such as the Web Player or Automation Services. Install TERR service on its own node.

Packages installed on a node manager running a different operating system than the engine

This unlikely case can happen if, for example, you are running the almalinux:8.9 Docker image for your TERR service, but your node manager computer is running Red Hat Enterprise Linux (RHEL) 9. Because of symbol version issues, packages must be installed in an equivalent OS to the execution environment. If this is not possible, try one of the following workarounds.

- Install the packages on a computer running the same operating system as your execution environment, and then copy those packages to the node manager running the other operating system. Provide the path in `custom.properties` file for the configuration `packagepath`.
- Create a customized image with almalinux:8.9, and then install the required packages in the image. Provide the image name in the `custom.properties` file in the configuration setting `docker.image.name`.

Error message "Bad Request (400) - Unable to parse form content"

If you see this error, then the data being uploaded to the service is too large.

When a data function is run, it sends the relevant data from Spotfire to TERR service for processing. TERR service has a file size upload limit, the default of which is 100 MB. When a data function is run with data that exceeds this limit, TERR service returns the following error:

```
Error from Remote Service: Bad Request (400) - Unable to parse form content
```

```
  at Spotfire.Dxp.Data.DataFunctions.Executors.RemoteServiceClient.RunFunction(DataFunctionInvocation
invocation)
  at
Spotfire.Dxp.Data.DataFunctions.Executors.SPlusFunctionExecutor.<ExecuteFunction>d__11.MoveNext()
  at Spotfire.Dxp.Data.DataFunctions.DataFunctionExecutorService.<ExecuteFunction>d__8.MoveNext()
```

An administrator can change this upload limit if needed by creating a custom configuration.

If users see this error message, consider creating a custom configuration that increases the limits for both of the following properties:

Property setting	Description
`spring.servlet.multipart.max-file-size`	The total file size for upload cannot exceed the value for this setting.
`spring.servlet.multipart.max-request-size`	The total request size for a multipart file upload cannot exceed the value for this setting.

At a certain point, the Java Heap size fails to handle very large files. The threshold for this behavior is dependent on the amount of RAM TERR service node has. When the Java Heap Size is exceeded, the following error message is displayed.

```
Error from Remote Service: Internal Server Error (500) - java.lang.OutOfMemoryError: Java
 heap space
```

Spotfire Documentation and Support Services

For information about the Spotfire® products, you can read the documentation, contact Spotfire Support, and join the Spotfire Community.

How to Access Spotfire Documentation

Documentation for Spotfire and TIBCO products is available on the TIBCO Product Documentation website, mainly in HTML and PDF formats.

The website is updated frequently and is more current than any other documentation included with the product.

Spotfire Documentation

The documentation for all Spotfire products is available on the Spotfire Documentation page. This page takes you directly to the latest version of each document.

To see documents for a specific Spotfire product or version, click the link of the product under 'Other versions', and on the product page, choose your version from the top right selector.

Release Version Support

Some release versions of Spotfire products are designated as long-term support (LTS) versions. LTS versions are typically supported for up to 36 months from release. Defect corrections will typically be delivered in a new release version and as hotfixes or service packs to one or more LTS versions. See also https://spotfi.re/lts.

How to Contact Support for Spotfire Products

You can contact the Support team in the following ways:

- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the support portal at https://spotfi.re/support.
- For creating a Support case, you must have a valid maintenance or support contract with Cloud Software Group, Inc. You also need a user name and password to log in to https://spotfi.re/support. If you do not have a user name, you can request one by clicking **Register** on the website.

System Requirements for Spotfire Products

For information about the system requirements for Spotfire products, visit https://spotfi.re/sr.

How to join the Spotfire Community

The Spotfire Community is the official channel for Spotfire customers, partners, and employee subject matter experts to share and access their collective experience. The Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from Spotfire products. In addition, users can submit and vote on feature requests from within the Ideas Portal. For a free registration, go to https://spotfi.re/community.

Legal and Third-Party Notices

Addendum to Legal and Third-Party Notices

Index