precisely

Spectrum Technology Platform

Spectrum Spatial Administration Guide

Version 2020.1.0

Table of Contents

8 - Troubleshooting Your System

1 - Introduction

In this section

What's Included in This Guide

Welcome to the *Spectrum Spatial Administration Guide*. This guide will help you build a web mapping application or embed mapping in an existing application using a variety of web services, capabilities, tools and sample code.

Addressed in this guide are:

- Configuring your system by changing the default port number or repository database; accessing the repository; accessing and uploading resources; configuring web services; and running Spectrum Technology Platform as a Linux service
- Managing security using the Management Console, including how to add users and roles, as well as how to apply security entity overrides
- Monitoring your system, including logging, viewing version and license information, using the Spectrum JMX Console to monitor performance, and monitoring memory usage
- Managing memory and threading, including JVM performance tuning, adjusting pool size, and increasing heap memory
- Load balancing spatial services for resilience or high capacity
- Troubleshooting your system, including rebuilding a corrupt repository index and monitory memory usage of a non-responsive server

Additional Spectrum Technology Platform and Spectrum Spatial documentation is located online at **support.precisely.com**.

2 - Configuring Your System

In this section

Changing the HTTP Port Number for Spectrum Spatial

The HTTP port is used to access all Spectrum Technology Platform web services, whether via REST or SOAP, and for the Welcome page, sample apps and Spectrum Spatial Manager.

After Spectrum Technology Platform is installed, you can change the existing port settings that were assigned during installation by manually editing the global, startup, and individual service configuration files. There are several reasons you may need to change the port number:

- A port conflict occurs after installation.
- You want to try out a new version of Spectrum Technology Platform without removing your old one. Since you cannot install them both, you can turn off the existing version and install a Spectrum Technology Platform image that uses a different port.
- You need a proxy on port 8080 but have a limited number of ports to expose externally, so you would like to move Spectrum Technology Platform without re-creating all your settings and data flows.

> **Note:** This task is only for experienced administrators who have application server experience changing port numbers, as network port conflicts can result in module components failing to start. One indication that a component has failed to start is if it does not appear in the Management Console. To troubleshoot the problem, look at the Spectrum Technology Platform server wrapper log. This log shows which port is causing the problem. You can find the wrapper log in: `<install_folder>\server\logs\spectrum-server.log`.

To change the port number:

1. In spectrum-container.properties change the value of `spectrum.http.port` to the new port number. This file is located in `<install_folder>/server/conf`.
2. In Spectrum Spatial Manager, change the port numbers in these service configurations:
   - Mapping (only necessary when accessing the Mapping Service via SOAP and when the ReturnImage parameter for a RenderMap request is False)
   - WFS
   - WMS
   - WMTS

   For instructions, see *Spectrum Spatial Manager* under *Managing Spectrum Spatial* in the *Spectrum Spatial Guide*.

   If you are relocating the server so it can use a different port, it is likely that the Spectrum Technology Platform server is not running. You will not be able to edit the service configuration

files until the server is running. You will need to start the server, edit the configurations and restart the server.

3. Restart Spectrum Technology Platform so the ports and property changes can take effect.

Changing Your Repository Database Type

Spectrum Spatial stores named resources (maps, layers, tables and styles), geographic metadata and configuration in a repository. In the default single server installation an embedded database is used to store these resources on the local server. There are several reasons you may need to use a database other than the embedded Derby database:

- To create a scalable solution that uses a resilient independent database.
- To use an in-house database preferred or dictated by your company.

In this release, the supported repository databases are Oracle, PostGreSQL/PostGIS, and Microsoft SQL Server. For instructions, see **Setting Up a Common Repository Database** on page 38.

Configuring the Web Services

You can, and frequently must, explicitly specify the desired behavior of the Spectrum Spatial web services via settings in each web service's configuration file. The configuration files for web services in Spectrum Spatial are held in the Spectrum Spatial repository as named configuration.

> **Note:** Named configurations are not like other named resources that are held in the repository. You cannot use the Named Resource Service to access named configurations. Instead, you must use a WebDAV tool such as WebFolders.

Configuration files are pre-loaded in the repository for the Mapping, Feature, Map Tiling, WFS, WMS, and WMTS services. These configuration files are located at:

```
http://hostname:port/RepositoryService/repository/default/Configuration/
```

For information about the name and location of each web service's named configuration in the repository, as well as a list of the configuration parameters for each web service, refer to the "Working With Spatial Services" chapter in the *Spectrum Spatial Developer Guide*.

Controlling Geometry Node Representation

The Spatial and Routing Module provide a property that allows to you control the number of digits that represent geometry nodes returned in a web service response. By default, geometries are returned without placing a limit on the number of digits, which could be as many as 16 digits long. The extra digits add unnecessarily to the payload of a JSON or SOAP response, particular when large polygons or many records are returned. It also has the potential of setting an expectation of accuracy that is not in the data. The difference of one in the least significant digit might be a value of a billionth of a meter. For example, 3989657.014543291 and 3989657.014543292 differ by one billionth of a meter. Spatial data rarely has anything close to that accuracy. By setting the property to true, the values are trimmed typically to 9 or 10 significant digits. Using the previous example, the value would be returned as 3989657.01 which has an accuracy of a centimeter.

To trim the node values, add the following property to %Spectrum%\server\bin\wrapper\wrapper.conf and restart the server.

```
wrapper.java.additional.xx=-Dcom.pb.midev.service.output.geometry.useprecision=true
```

where `xx` is the number of the next available line in the section.

The coordinate values will be handled the same way for all geometries across services, whether for SOAP or REST calls, including services exposed from a data flow. This includes Spectrum Spatial Feature Service, Mapping Service, Geometry Service, Map Tiling Service, WMS, WMTS, and WFS and the routing services.

Applications that are editing polygon data through the web services should not use this property if it is possible that by writing back trimmed geometries, small overlaps or gaps might be created with neighboring geometries.

Disabling Accuracy Files for Datum Transforms

Spectrum Spatial supports conversions between certain datums by using algorithms that help shift coordinates more accurately. A separate jar file that contains these algorithms is installed by default for each datum transform located in the *Spectrum Installation Location*\server\types directory:

- midev-core-coordsys-irishtm-*version number*-onprem.jar for Irish Transverse Mercator
- midev-core-coordsys-jgd2000-*version number*-onprem.jar (also enables the updated version, JGD2011) for Japanese datums
- midev-core-coordsys-nadcon-*version number*-onprem.jar for US Nad27-Nad83

- midev-core-coordsys-ntv2-*version number*-onprem.jar for NTV2, which contains multiple conversions for many countries.

  Note: An XML file inside this jar controls which conversions are in use. To disable specific conversions within that file, stop the server and extract the XML file from the jar. Use an editor to set the entries to "false" for each conversion you want to disable. Add the edited XML file back into the jar, then restart the server. Similarly, if you want to enable the conversion, set the entries to true. For details see **Enabling NTv2 Transform**.

- midev-core-coordsys-rgf93-*version number*-onprem.jar for French Lambert conversions

By default, all of these jar files are loaded; however, their use can negatively affect the performance of certain operations. These conversions can be disabled in some cases, such as when you do not require a certain type of conversion (for example, if you have no need to convert Japanese datums) or the performance gains outweigh the benefits of accuracy at lower zoom levels.

To disable a specific transform:

1. Stop the server.
2. Remove the jar from the directory. Alternatively, you can rename the jar file to have a different extension (for example, .jar~) which will prevent it from being loaded.
3. Restart the server.

Configuring Request Timeouts

Spectrum Spatial lets you set a timeout for SOAP and REST operations as part of a request to the Mapping and Feature services.

Slow requests may take a long time to finish and can consume server resources so that the server might not respond to other requests. You can set a timeout for SOAP and REST requests to the Mapping and Feature services to protect the server from overload. When the timeout is enabled (timeout > 0 seconds), Spectrum Spatial monitors requests and terminates them when they do not finish during the timeout period. During termination, Spectrum Spatial cancels processing threads and releases resources (CPU and Memory).

The timeout feature is disabled by default. To set these timeout features, see Setting Java Properties.

Properties Name	Default Value (in seconds)
timeout.feature.value	-1
timeout.mapping.value	-1

A negative value (such as -1) disables the timeout feature.

Configuring a Query Timeout

When terminating a request that is accessing a database, the query may continue to run in the database. To prevent this from happening, you can set up a query timeout set to the same value as the request timeout. The query timeout terminates the query in the database when there is a request timeout. The following example shows a query timeout value of 30 seconds.

Properties Name	Value (in seconds)
pool.database.jdbcInterceptors	QueryTimeoutInterceptor(queryTimeout=30)

Configuring the Volatile Attribute for Named Tables

Volatility is an indication to Spectrum Spatial that information from a data source can change at any time. The default value for TAB, and JDBC-based (Oracle, SQL Server and PostGIS) named tables is set to true, meaning that for each data access operation, such as a query or insert, Spectrum Spatial checks with the data source to find out if the table is volatile and if so, whether the data changed. If the data has changed, the cache is flushed and the table is reloaded before the data access operation can proceed. If the table did not change, the query or other operation is carried out on the data in the cache. See **Supported Data Sources** in the *Spectrum Spatial Guide* for more information about what triggers a change for each data source.

Volatility is disabled (set to false) for named tables that are uploaded from MapInfo Pro using **Map Uploader**. Volatility is enabled for any named tables created with **Spectrum Spatial Manager**. Older named tables in the repository are considered to be volatile but will not indicate that when viewed in the Spectrum Spatial Manager table details page.

Disabling volatility should be done only on tables that do not change. For example, when generating tiles from volatile TAB files, the operation will perform very slowly. If you are using PostGIS, you may also want to consider disabling volatility to avoid encountering connection errors in Spectrum Spatial Manager (for example, when viewing the sample rows on the table details page).

Volatility can be disabled on the table details page in Spectrum Spatial Manager. For more information on creating and modifying named tables in Spectrum Spatial Manager, see the *Managing Spatial* section of the *Spectrum Spatial Guide*.

You must **restart Spectrum Spatial in Spectrum Spatial Manager** when the named table is set as non-volatile.

> **Note:** Do not use the updateNamedResource operation in the Named Resource Service to change this value or manually edit the named table definition that you accessed via WebDAV in a text editor.

Running Spectrum Technology Platform as a Linux Service

This tutorial will show you the steps you need to follow to run Spectrum Technology Platform as a Linux service.

How to Run Spectrum Technology Platform as a Linux Service

These instructions describe how to run the Spectrum Technology Platform as a Linuxservice.

1. Modify the provided `pbspectrum` script which is located here: **Spectrum Script** on page 13.
   a) Modify the `chkconfig` parameter at line# 5. By Default this parameter is: `# chkconfig: 35 90 10`

      First value(35) is runlevel. Use 'man init' for more information.

      Second value(90) is start priority

      Third value(10) is stop priority.

      Start and stop priority should be set according to the dependent services. For example, if Oracle Server is running on the same machine and is used by Spectrum Technology Platform then the Spectrum Technology Platform starting priority should be less than the Oracle Service and stopping priority should be higher than the Oracle service. Use 'man chkconfig' for more information.

   b) Modify SPECTRUM_ROOT variable at line #11 with your Spectrum Technology Platform installation directory.
   c) If you are using SUSE Linux, you must change the default preferred user from `su` to `runuser`.
2. Copy the modified `pbspectrum` script to either `/etc/rc.d/init.d` for RedHat Linux or `/etc/init.d` for Suse Linux.
3. Change the mode of the `pbspectrum` script to executable. `/etc/rc.d/init.d` for RedHat Linux or `/etc/init.d` for Suse Linux.

`cd /etc/init.d` or `cd /etc/rc.d/init.d` depending on your Linux version.

run `chmod +x pbspectrum`

4. Run `chkconfig --add pbspectrum`
5. Verify the script is working by restarting the machine. Use `shutdown -r now` to reboot from shell.

Once completed, you may also use the following:

- `service pbspectrum start` to start Spatial Server
- `service pbspectrum stop` to stop Spatial Server
- `service pbspectrum restart` to restart Spatial Server

Note: The provided script runs the command 'ulimit -n 8192' which is required to increase the number of open files in Linux.

Spectrum Script

The following script is used as the basis for the procedure described under **How to Run Spectrum Technology Platform as a Linux Service** on page 12.

Note: For Linux, the default Spectrum Technology Platform install folder is `/home/<user>/Spectrum`.

```
   #! /bin/bash
#
#  spectrum Bring up/down the Spectrum platform
#
# chkconfig: 35 90 10
# description: Starts and stops the spectrum
#
# /etc/rc.d/init.d/spectrum
# See how we were called.

SPECTRUM_ROOT=/root/Spectrum

start() {
        su - spectrum -c ". $SPECTRUM_ROOT/server/bin/setup;
  ulimit -n 8192;
  $SPECTRUM_ROOT/server/bin/server.start"
        RETVAL=$?
        return $RETVAL
}

stop() {
```

```
        su - spectrum -c ". $SPECTRUM_ROOT/server/bin/setup;
        $SPECTRUM_ROOT/server/bin/server.stop"
        RETVAL=$?
        return $RETVAL
}

# See how we were called.
case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  restart)
        stop
        start
        ;;
  *)
        echo $"Usage: spectrum {start|stop|restart}"
        exit 1
esac

exit $RETVAL
```

Configuring a Linux Machine for MRR

To use MRR (Multi Resolution Raster) files on Spectrum Spatial in a Linux environment, GCC and LIBC must be upgraded to the proper versions.

To configure a Linux machine for MRR:

1. Install the UUID package, which installs LIBC v.2.17.

   For example, to install UUID on Cent OS:

   - `wget http://ftp.riken.jp/Linux/centos/6/os/x86_64/Packages/libuuid-2.17.2-12.18.el6.x86_64.rpm`
   - `sudo yum -y install libuuid-2.17.2-12.18.el6.x86_64.rpm`
   - `sudo yum -y install libuuid-devel`

2. Install devtoolset-3, which installs GCC v.4.9. For instructions, see **https://www.softwarecollections.org/en/scls/rhscl/devtoolset-3/**.
3. Verify that GCC v.4.9 and LIBC v.2.17 (or higher) are installed.

4. Ensure that all the dependencies were resolved in the above steps. If any dependency is unresolved, install it and then repeat Step 2.

   For example, the following are some of the required dependencies for an OEL 6.5 machine:

   - `wget https://www.softwarecollections.org/en/scls/mizdebsk/maven30-rhel-6/epel-6-x86_64/download/mizdebsk-maven30-rhel-6-epel-6-x86_64.noarch.rpm`
   - `sudo yum -y install mizdebsk-maven30-rhel-6-epel-6-x86_64-1-2.noarch.rpm`
   - `wget https://www.softwarecollections.org/en/scls/rhscl/maven30/epel-6-x86_64/download/rhscl-maven30-epel-6-x86_64.noarch.rpm`
   - `sudo yum -y install rhscl-maven30-epel-6-x86_64-1-2.noarch.rpm`
   - `sudo yum -y install maven30`
   - `wget https://www.softwarecollections.org/en/scls/mbooth/eclipse-luna/fedora-20-x86_64/download/mbooth-eclipse-luna-fedora-20-x86_64.noarch.rpm`
   - `sudo yum -y install mbooth-eclipse-luna-fedora-20-x86_64-1-2.noarch.rpm`
   - `sudo yum -y install --skip-broken eclipse-luna`

Exposing Cache Control Headers for Map Display

By default, Spectrum Technology Platform web services use the following HTTP headers for caching:

```
Cache-Control: no-cache,no-store,no-transform,must-revalidate
Expires: Wed, 07 Jan 2015 15:38:03 GMT //48 hours in the past
Pragma: no-cache
```

These HTTP headers are not appropriate for the Map Tiling Service. You can disable these default HTTP headers and set the HTTP cache behavior in the headers that are defined in the individual web services instead.

> **Note:** If you are applying this change to a cluster you must repeat the following procedure on each node in the cluster.

To disable the default HTTP cache control headers:

1. Stop the Spectrum Technology Platform server.
2. In a text editor, open the following file:

   SpectrumDirectory`\server\conf\spectrum-container.properties`

3. Change the following property from true to false:

```
spectrum.http.cache.control.headers.enable=false
```

4. Save and close the properties file.
5. Start the Spectrum Technology Platform server.

3 - Monitoring Your System

In this section

Viewing System Events

View the system log when you experience trouble and are looking for information about possible causes.

This procedure downloads the current system log `spectrum-server.log` from the Spectrum Technology Platform server. If you are running Spectrum in a clustered environment, the system log file downloaded by this procedure is the current log file from the node to which you are connected.

1. On the Spectrum Technology Platform home page, click **Platform Client Tools** > **Web** > **Open Management Console**.
2. On the **Sign in** page, enter your credentials.
3. On the **Management Console** page, click **System** > **Logs**.
4. On the **System Log** tab, click the **Download log file** button to download the system log file.
5. Open the downloaded file in a text editor.

Setting Logging Levels for Services

You can specify the default logging level as well as logging levels for each service on your system. When you change logging levels the change will not be reflected in the log entries made before the change.

> **Note:** The logging levels you specify for services do not affect the audit log. They only control the level of logging for the event log which you can view in Spectrum Management Console. At this time you cannot view the event log in the web version of Spectrum Management Console.

1. Open Spectrum Management Console.

   On the **Home** page, click **Platform Client Tools**, expand **Web**, then click **Open Management Console**.
2. Click **System** > **Logs**.
3. Click the **System Log** tab.
4. In the **System default logging level** box, select a default event logging level for services on your system.

Disabled	No event logging enabled.
Fatal	Minimal logging. Only fatal errors are logged. Fatal errors are those that make the system unusable.

Error	Errors and fatal errors are logged. Errors indicate an isolated problem that causes part of the system to become unusable. For example, a problem that causes a single service to not work would generate an error.
Warn	Event warnings, errors, and fatal errors are logged. Warnings indicate problems that do not stop the system from working. For example, when loading a service where a parameter has an invalid value, a warning is issued and the default parameter is used. During the use of a service, if results are returned but there is a problem, a warning will be logged.
Info	High-level system information is logged. This is the most detailed logging level suitable for production. Info events are typically seen during startup and initialization, providing information such as version information and which services were loaded.
Debug	A highly detailed level of logging, suitable for debugging problems with the system.
Trace	The most detailed level of logging, tracing program execution (method entry and exit). It provides detailed program flow information for debugging.

Each logging level includes the ones above it on the list. In other words, if Warning is selected as the logging level, errors and fatal errors will also be logged. If Info is selected, informational messages, warnings, errors, and fatal errors will be logged.

> **Note:** Selecting the least severe and therefore most verbose logging level can affect system performance. We therefore recommend that you should select the most severe setting that meets your particular logging requirements.

5. If you want to specify a different logging level for any service, choose the logging level in the **Logging Level** column of the table.

Spatial Logging

The logback.xml file allows you to control on logging behavior, such as sending output to a log file instead of by default sending it to the console which redirects to the spectrum-server.log. You can also set the log level to turn off logging altogether or log only fatal errors, for example.

Default logback file

(`<Installed>\Precisely\Spectrum\server\modules\spatial\logback.xml`)

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
```

```
============================================================================================
 -->
<!-- Logger configuration for remote components
                    -->
<!--
                    -->
<!-- log to console, redirected to Platform log
(server\logs\spectrum-server.log) -->
<!-- log to files, redirected to (server\modules\spatial\spatial.XXX.log)
                  -->
<!--
                    -->
<!-- for general information about the configuration file, check out the
 logback manual  -->
<!-- at http://logback.qos.ch/manual/configuration.html
                    -->
<!--
============================================================================================
 -->
<configuration>
 <appender name="CONSOLE-SPATIAL"
class="ch.qos.logback.core.ConsoleAppender">
  <encoder>
   <pattern>[Spatial] - [%thread] %-5level %logger{35} - %msg%n</pattern>

  </encoder>
 </appender>
 <!--appender name="FILE-SPATIAL"
class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>${g1.server.modules.dir}/spatial/${component.name}.log</file>
  <encoder>
   <pattern>%d [%thread] %-5level %logger{35} - %msg%n</pattern>
  </encoder>
  <append>true</append>
  <triggeringPolicy
class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
   <maxFileSize>10MB</maxFileSize>
  </triggeringPolicy>
  <rollingPolicy
class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
   <fileNamePattern>${component.name}.log.%i</fileNamePattern>
   <maxIndex>1</maxIndex>
  </rollingPolicy>
 </appender-->
 <!-- Level: OFF, ERROR, WARN, INFO, DEBUG -->
 <logger name="com.mapinfo.midev" level="INFO" additivity="false">
  <appender-ref ref="CONSOLE-SPATIAL"/>
  <!-- appender-ref ref="FILE-SPATIAL"/ -->
 </logger>
</configuration>
```

Option	Values
Level	• OFF–turn off logging • ERROR–log runtime or unexpected errors • WARN–log warnings only; for example, using a deprecated API • INFO–log runtime events such as startup or shutdown [default] • DEBUG–log detailed debugging information
Output	• CONSOLE-SPATIAL –sends log information to the console [default] • FILE-SPATIAL–sends log information to a log file based on component (no longer applicable - Spectrum Spatial has a single remote component)

Configuring a Mail Server

Spectrum Technology Platform can send email alerts to notify you of important events. Email notifications can be sent as a result of conditions within dataflows and process flows, and when time-based licenses, databases, and other items are about to expire.

Spectrum Technology Platform does not have a built-in mail server, so in order to enable email notification you must configure it to use an external SMTP server.

1. Open Spectrum Management Console.
2. Navigate to **System** > **Mail Server**.
3. In the **Host** field, enter the host name or IP address of the SMTP server you want to use to send email notifications.
4. In the **Port** field, enter a port number or range to use for network communication between the Spectrum Technology Platform server and the SMTP server.

   The default port is 25.
5. In the **User name** and **Password** fields, enter the credentials that the Spectrum Technology Platform server should use to authenticate with the SMTP server.
6. In the **From address** field, enter the notification sender's email address.
7. To confirm that you have correctly configured a mail server, you can send a test email. Enter the email address you want to send the test to in the **Test address** field then click **Test**.

8. Click **Save**.

The Spectrum Technology Platform server is now connected to an SMTP server and can use that server to send notification email.

Example: Configuring a Mail Server

You have an SMTP server named `mail.example.com`. You want to use this mail server to handle email notifications sent from the Spectrum Technology Platform server. You have created an account on the SMTP server called Spectrum123 with a password of Example123, and the email address for this account is `spectrum.notification@example.com`.

To configure notification with this information, you would complete the fields as follows:

Host	`mail.example.com`
From address	`spectrum.notification@example.com`
User name	`Spectrum123`
Password	`Example123`

Related concepts

Notifications

Selecting Items for Expiration Notification

You can choose which items you want to be notified about so that you only receive notifications for those items that concern you.

Spectrum Technology Platform can send an email notification when a license, database, or software component is about to expire. This allows you to take the necessary action to ensure that your business processes are not disrupted by an expiration. Some of the components that have expiration dates include:

- Licenses
  - Email notifications are not available for transaction-based licenses. If you are approaching the maximum number of transactions for a license, a message appears in the system log in Spectrum Management Console.
  - When you log in as admin in Spectrum Spatial Manager, and the license expiry date falls inside the license expiration range set in Spectrum Management Console, a warning message is displayed as: `Spatial License will expire in <n> days.`
- Databases, such as U.S. postal databases used for CASS processing

- Certain software components, such as the engine used to validate U.S. addresses in Spectrum Universal Addressing.

Tip: To view the items that have expiration dates, open Spectrum Management Console and go to **System** > **Licensing and Expiration**.

1. Open Spectrum Management Console.
2. Go to **System** > **Licensing and Expiration**.
3. To receive an expiration notification email for an item, check the box in the **Send Notification** column.

   If you want to be notified earlier or later than the default, specify the number of days in advance of the expiration that you want to be notified.

Viewing Version Information

1. In a web browser go to this URL:

   `http://server:port/managementconsole`

   Where *server* is the server name or IP address of your Spectrum Technology Platform server and *port* is the HTTP port used by Spectrum Technology Platform. By default, the HTTP port is 8080 and the HTTPS port is 8443.

2. Click **System** > **Version**.

Viewing and Exporting License Information

You can export information about your license to an XML file. This may be necessary when resolving license issues with technical support.

1. In a web browser go to this URL:

   `http://server:port/managementconsole`

   Where *server* is the server name or IP address of your Spectrum Technology Platform server and *port* is the HTTP port used by Spectrum Technology Platform. By default, the HTTP port is 8080 and the HTTPS port is 8443.

2. Click **System** > **Licensing and Expiration**.
3. Click the export icon.

Your license information is saved to an XML file with a `.lic` extension.

Monitoring Performance with the Spectrum JMX Console

The Spectrum JMX console is a browser-enabled tool that provides a performance monitoring tool that records performance statistics for each stage in a dataflow.

1. Open a web browser and go to `http://server:port/jmx-console`

   Where:

   server is the IP address or host name of your Spectrum Technology Platform server.

   port is the HTTP port used by Spectrum Technology Platform. The default is 8080.

2. Log in using the admin account.
3. Under **Domain: com.pb.spectrum.platform.performance**, click **com.pb.spectrum.platform.performance:service=PerformanceMonitorManager**.
4. Click the **Invoke** button next to **enable**.
5. Click **Return to MBean View** to go back to the PerformanceMonitorManager screen.

Performance monitoring is now enabled. When a dataflow runs, the performance statistics will display at the top of the PerformanceMonitorManager screen. Note the following:

- You must refresh the screen to see updates.
- To reset the counters, click the **Invoke** button next to **reset**.
- If you stop the Spectrum Technology Platform server, performance monitoring will be turned off. You will have to turn it back on when you start the server again.

Monitoring File Handle Caching Statistics with the Spectrum JMX Console

The Spectrum JMX console is browser-based tool that monitors performance and records statistics, including file handle caching statistics for native TAB and shapefiles.

1. Open a web browser and go to `http://server:port/jmx-console`

   Where:

server is the IP address or hostname of your Spectrum Technology Platform server.

port is the HTTP port used by Spectrum Technology Platform. The default is 8080.

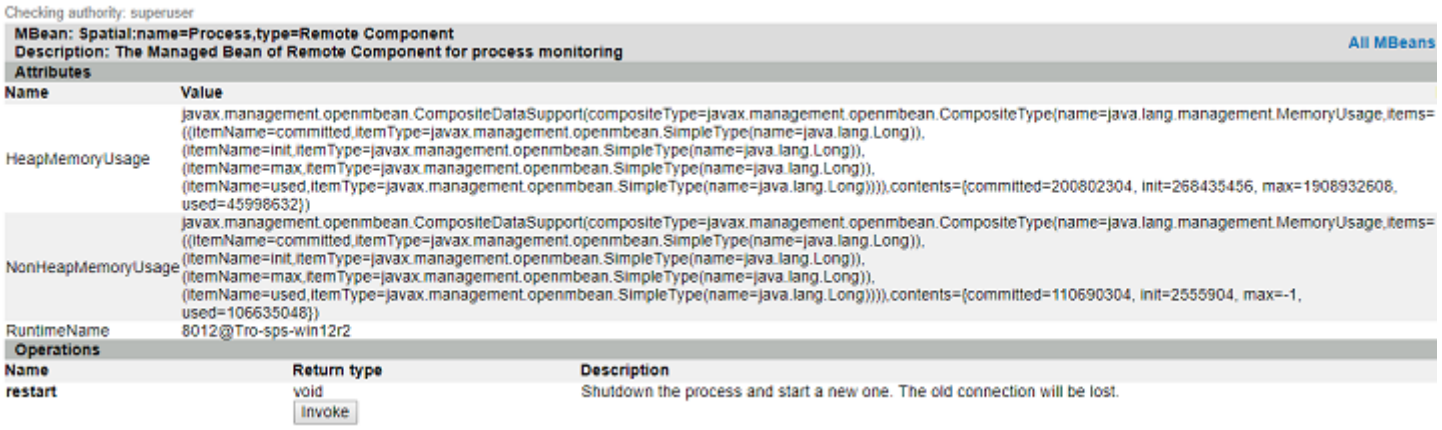
2. Log in using the admin account.
3. Under " Domain: Spatial", click **Spatial:name=TABFileHandlePool,type=Remote Component** or **Spatial:name=ShapeFileHandlePool,type=Remote Component** to view the file handle caching statistics for TAB files or shapefiles.

   Note: You can also disable the file handle cache or clear it on this page without needing to restart the server.

4. Click **All MBeans** to return to the main Spectrum JMX Console.

Monitoring Memory Usage

The Spectrum JMX Console allows you to monitor the JVM heap usage of the spatial remote component.


Checking authority: superuser

MBean: Spatial:name=Process,type=Remote Component All MBeans
Description: The Managed Bean of Remote Component for process monitoring

Attributes

Name	Value
HeapMemoryUsage	javax.management.openmbean.CompositeDataSupport(compositeType=javax.management.openmbean.CompositeType(name=java.lang.management.MemoryUsage,items=((itemName=committed,itemType=javax.management.openmbean.SimpleType(name=java.lang.Long)),(itemName=init,itemType=javax.management.openmbean.SimpleType(name=java.lang.Long)),(itemName=max,itemType=javax.management.openmbean.SimpleType(name=java.lang.Long)),(itemName=used,itemType=javax.management.openmbean.SimpleType(name=java.lang.Long)))),contents={committed=200802304, init=268435456, max=1908932608, used=45998632})
NonHeapMemoryUsage	javax.management.openmbean.CompositeDataSupport(compositeType=javax.management.openmbean.CompositeType(name=java.lang.management.MemoryUsage,items=((itemName=committed,itemType=javax.management.openmbean.SimpleType(name=java.lang.Long)),(itemName=init,itemType=javax.management.openmbean.SimpleType(name=java.lang.Long)),(itemName=max,itemType=javax.management.openmbean.SimpleType(name=java.lang.Long)),(itemName=used,itemType=javax.management.openmbean.SimpleType(name=java.lang.Long)))),contents={committed=110690304, init=2555904, max=-1, used=106635048})
RuntimeName	8012@Tro-sps-win12r2

Operations

Name	Return type	Description
restart	void Invoke	Shutdown the process and start a new one. The old connection will be lost.

Memory usage (HeapMemoryUsage and NonHeapMemoryUsage) is based on the standard JVM memory MBean. It shows the memory usage of the JVM that the remote component running on. It includes the amount of init, max, committed and used memory.

RuntimeName includes the process ID that you can use to find more information from the operating system (for example, by using the Windows Task Manager), or even kill the process.

In the heap sections, `={committed=200802304, init=268435456, max=1908932608, used=45998632}` are shown in bytes.

You can modify the heap memory in the Management Console. For details, see **Increasing Heap Memory for the Spatial Remote Component**.

Restarting Spectrum Spatial

You can use the Spectrum Spatial Manager to restart Spectrum Spatial remote component to clear the cache or pick up new settings without restarting the Spectrum Technology Platform.

How to Restart Spectrum Spatial

1. Launch Spectrum Spatial Manager.
2. Click **Settings** and then open the **Settings** sub-menu.
3. Click **Re-start** from **Re-start Spectrum Spatial**. A warning message displays.

   Note: Some requests can fail during restart, so we recommend restarting when there is no traffic.

4. Click **Yes** to restart.

Cases where Spectrum Spatial Restart is Helpful

1. Spectrum Spatial caches table structures, geometry, and index-related information that does not modify frequently. The data in the tables is mostly static, but may refresh occasionally. When deploying new data, we recommend restarting Spectrum Spatial to clear the cache. You can update a table's volatile setting from the Modify Table capability in Spectrum Spatial Manager.

   To understand the volatility of a data source, **click here**.

   Note:

   - For file-based tables, like tab or shape files, restart the Spectrum Spatial when the table structure or data has changed.
   - In the case of database tables, restart the Spectrum Spatial only when the structure of the table has been modified.

Clearing MRR Cache

The MRR file is locked as long as its handle is open in the cache, therefore preventing any update, delete, or replace operations on the file. To release the MRR native handle, an option is available on the Spectrum JMX Console to manually close all open handles:

1. Open a web browser and go to `http://server:port/jmx-console`

Where:

server is the IP address or hostname of your Spectrum Technology Platform server.

port is the HTTP port used by Spectrum Technology Platform. The default is 8080.

2. Log in using the admin account.
3. Under "Domain: Spatial", click **Spatial:name=MRRCache,type=Remote Component**.
4. Click the **Invoke** button next to the **closeAll** operation to close all the open MRR handles.

You will get a message on the status of the invocation.

Note: A forced closure can throw an exception when the handle is being used in parallel for the purpose of rendering. Such exceptions are ignored but logged.

4 - Performance Tuning

This section describes approaches for improving performance by managing memory and threading, and also relates best practices for optimizing the performance of Spectrum Spatial. It is intended for experienced administrators.

Spectrum provides several tuning options to optimize performance of the server. The optimal selection of settings is dependent on the nature of the deployment. To create a well-tuned server environment, it is recommended that performance tests should be executed in the deployed environment to determine optimal settings. This section provides some general guidance on performance tuning.

In this section

Remote Component Configuration

All spatial services in the Spectrum Technology Platform are deployed into a remote component (JVM instance) that is separate from the platform runtime. This ensures the platform is independent of the modules within it and that JVM configuration can be applied to the spatial services, allowing flexibility of memory allocation and tuning for performance based on the characteristics of those services.

The remote component supplies spatial functions to spatial services (such as the Feature Service and Mapping Service) and stages (such as the Spatial Calculator and Query Spatial Data). The pool size for a remote component is the number of requests the component can handle concurrently. This affects the throughput of both spatial services and spatial stages.

Modifying the Pool Size

In addition to JVM tuning, you can also adjust the pool size of the spatial remote component. The pool size for a remote component is the number of requests the component can handle concurrently. This setting represents the number of threads on the components that are listening for service requests from the Spectrum Technology Platform or executing a Spectrum Spatial stage (that is, the maximum number of managed connections).

Every web service request enters Spectrum from the platform and is passed to the component. The default value of 4 can be increased to accommodate greater request loads. A pool size that matches the number of CPUs is recommended. The maximum setting should not go above twice the number of the CPU core; for example, on a 4 CPU machine the combined number of threads for all services should not exceed 8. Performance tests should be run with various settings until optimal performance is achieved for the usage.

You have the ability to adjust the pool size in Management Console for the spatial remote component:

1. Open the Management Console.
2. Go to **Resources** > **Spatial**.
3. Change the pool size for the remote component using the arrows or by typing in a value. The minimum value is 1 and the maximum value is 64.
4. Click **Save**.
5. If you decreased the pool size, restart the server. Increasing the pool size takes effect immediately and does not require a server restart.

Data Source Pooling Configuration

Properties for managing the pooling of database connections, used by JDBC-based data sources (such as Oracle and SQL Server) to optimize performance, are set by an administrator (admin) in the Management Console. For instructions about setting Spatial java properties, see **Setting Spatial Java Properties**. For details about *pool.database* settings, see **Spatial Java Properties**.

In most cases, we recommend enabling the connection validation by setting *pool.database.testOnBorrow* to *true*. This validates objects before borrowing them from the pool. If the validation fails, the connection is dropped from the pool and an attempt will be made to borrow another. A validation query is also available for special cases, such as when using a custom data provider. If the validation query is enabled, it is used.

Enabling validation may have a slightly negative performance impact; however, the test query maintains the integrity of all the connections in the connection pool in cases where communication between Spectrum Spatial and an external database is not reliable. Set a validation interval to mitigate the performance impact of validation. If a connection is due for validation but has been validated previously within this interval, it will not be validated again.

Improving Performance for Distance-Based Operations

A PGD index file is a supplemental file to the TAB file set that can make performance for native, native extended (NativeX), and seamless TABs comparable to that of GSB files. The PGD Builder, a command-line utility, is available to generate these specialized index files to improve performance of certain distance-based operations for native datasets containing lines and polygons. An index built using the PGD Builder is helpful when the data you are searching is based on lines and regions and you are using:

- the Point in Polygon stage, when you are including distance
- the Find Nearest stage, when the input is a point (whether or not you are including distance)
- SearchNearest operations in the Feature Service, with an input point and a line or polygon search table

The PGD Builder utility can be downloaded from the Spectrum Spatial section of the Welcome Page, under **PGD Builder** on the Utilities tab. A link to the is also available on the Welcome Page next to the download link for the utility.

Note: A PGD file is 5-6 times larger than the .MAP file for the TAB. One PGD file is generated per TAB file, except in the case of a seamless TAB which will have PGD files created for each sub-TAB.

Also, a PGD file will no longer be used by the system if you change the data in the TAB (that is, if rows have been added or deleted or a geometry has been changed in the MAP portion of the TAB). If warnings are enabled (see **Spatial Logging** on page 19), a message about the out-of-date PGD file will appear in the `spectrum-server.log` or if applicable, in the log file that has been configured for Spatial logging. You must then regenerate the PGD for the updated TAB file.

Managing ACL Properties

Disabling the following Access Control List (ACL) security checks in the Spectrum JMX Console results in faster processing and avoids slow server startup:

- **spatial.security.acl.enable** If the system administrator is not using permissions on any Named Resources in the Spatial repository, they can set this property to false to disable the ACL check. Since all users have full permissions on all Named Resources, setting this property to false skips the consistency checks between the Spatial repository and the access control database at system startup time. This results in faster processing.
- **spatial.security.acl.check** On system startup, Spatial runs a consistency check. This process can be time-consuming depending on the number of Named Resources in the Spatial repository. Administrators can set this property to false to avoid the check and thereby speed up the server startup time.

A Spectrum Technology Platform server restart is required after the change.

To disable ACL security checks:

1. Open a web browser and go to `http://server:port/jmx-console`

   Where:

   server is the IP address or hostname of your Spectrum Technology Platform server.

   port is the HTTP port used by Spectrum Technology Platform. The default is 8080.

2. Log in using the admin account.
3. Under "Domain: com.pb.spectrum.platform.configuration.properties", click **com.pb.spectrum.platform.configuration.properties:manager=spatial.properties**.
4. Change the value of **spatial.security.acl.enable** and/or **spatial.security.acl.check** to **false**.
5. Click the **Set** button next to the **spatial.security.acl** property that you changed.
6. Restart the server.

Once finished, the ACL security checks are disabled.

5 - Managing a Cluster

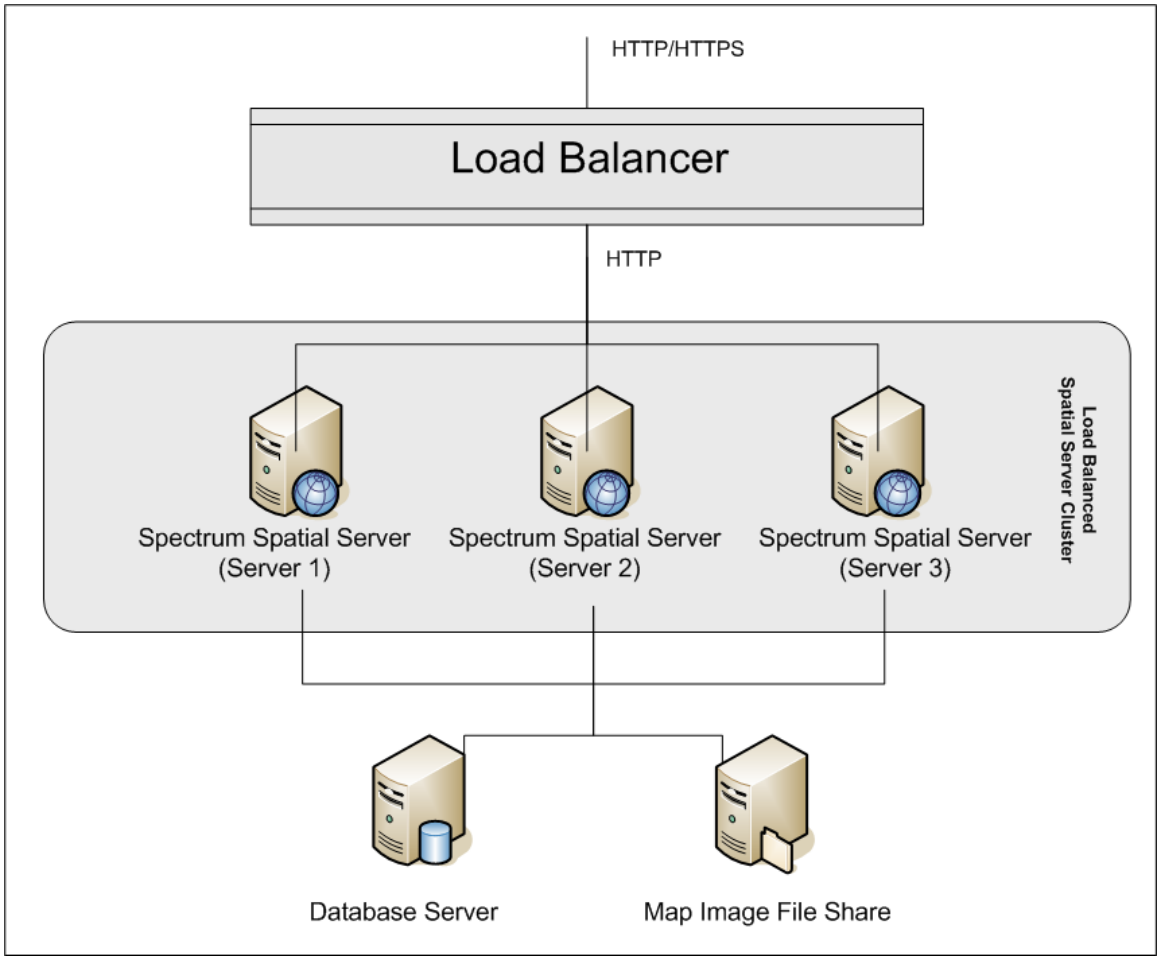
In this section

Clustered Architecture for Spatial

In a clustered environment, processing is shared among two or more instances of the server. The diagram below illustrates the deployment architecture of such a configuration. Load balancing can be used to support high availability and scaling. The deployment architecture includes a load balancer, a Spectrum Spatial cluster, a database, and a file share. With this approach it is possible to scale both horizontally and vertically. You can cluster Spatial with or without platform clustering.

Note: Setting up both a Spectrum Technology Platform cluster and a Spatial cluster is recommended and has several benefits:

- Security (ACL) synchronization happens automatically for named resources .
- Dataflows, users, and roles created on one node will automatically synchronize to all nodes.
- All Spatial demo pages and utilities (such as Spectrum Spatial Manager) can and should point to the load balancer.



Load Balancer

The load balancer spreads requests between the Spectrum Spatial instances. Any load balancer that supports load balancing HTTP/HTTPS requests can be used.

Spectrum Spatial Cluster

The cluster is a collection of Spectrum instances with Spatial sharing administration, named resources, geographical metadata content and configuration settings. Additional nodes can be added to the cluster for resilience or to deliver support for greater loads. Each node can be scaled vertically through additional hardware resources and/or additional instances should this be required for hardware with massive resources. Spectrum can be configured to use restricted numbers of CPUs.

Database

Spectrum stores named resources (maps, layers, tables and styles), geographic metadata and configuration in a repository. In the default single server installation an embedded database is used to store these resources on the local server. To create a resilient scalable solution this embedded database should be replaced with a resilient independent database. Oracle, PostGreSQL/PostGIS and Microsoft SQL Server are the supported repository databases.

In the load balanced configuration, Spectrum nodes cache these resources in a local cache and search index in each node in the cluster. When a Spectrum node receives a request it uses the local cache and index to find resources. Named resources can be added through any node in the cluster. Each node keeps its cache current by checking for differences between its local cache and the central database. This check occurs every 2 seconds by default. Time frequency can be configured. This architecture ensures the server delivers high performance transactions and the load on the repository database is kept to a minimum. If a new Spectrum node is added to the cluster the cache and index are created automatically. Such a scenario can occur to remedy a node failure or grow the capability of the deployment.

File Share

The file share provides a folder to hold map images generated by Spectrum. When maps are rendered using the web services the server supports the map images being returned through URLs or returned as a base 64 encoded image. When a URL is returned the map image is stored as a file and served on request of the URL. To ensure any Spectrum node can return the map image a file share is used to store the images.

Using Enterprise Designer with a Cluster

1. Launch Enterprise Designer.

2. In the **Server name** field, enter the server name of the load balancer.
3. In the **Port** field, enter the port that you have configured the load balancer to listen on.

   Note: Input files, output files and database resources must be on a shared drive, or file server, or some commonly-accessible location. Otherwise, all files must be loaded on each server that hosts a Spectrum Technology Platform server and must be located in the same path.

Once you have logged in you can use Enterprise Designer as normal. The actions you take will apply to all Spectrum Technology Platform instances in the cluster where you are logged in.

Starting a Cluster

These instructions assume that the server is stopped.

If all the nodes in a cluster are stopped, you must follow this procedure to start the cluster safely and avoid data loss.

On the last node that was stopped last, start the server. Do this for each node in the cluster.

Warning: The first node that you start must be the last node that was stopped to preserve the most recent data. Starting another node first may result in loss of data such as job history and configuration settings. If you do not know which node was stopped last, look in each node's log for the time stamp of the shutdown message. You can find the log in: *SpectrumDirectory*`\server\logs\spectrum-server.log`.

a) Start the server.
b) Start all nodes consecutively, after upgrading. Make sure that you start node 2 within only a few seconds after starting node 1, and repeat this for each remaining node.

   You can tell when the Spectrum Technology Platform server has completely started by looking in the log file: *SpectrumDirectory*`\server\logs\spectrum-server.log`. This message is displayed when the server is completely started:

```
Precisely Spectrum Technology Platform  (Version Version Number)
Started.
```

   The log will show the IP address for one of the nodes bound to the Spectrum Technology Platform service.

Stopping a Cluster

To stop an entire cluster:

1. Identify which nodes are seed nodes. To do this, open the file *SpectrumDirectory*`/server/conf/spectrum-container.properties` and look at the nodes listed in the `spectrum.cluster.seeds` property.
2. Stop each Spectrum Technology Platform server in the cluster, making sure that the last node you stop is a seed node.
3. Change the working directory to the Spectrum Technology Platform server's `bin` directory, source the setup file, then type the following command: `./server.stop`.

   Warning: To prevent loss of data when starting the cluster, the first node you start must be the last node that was stopped, and that node must be a seed node.

4. Make a note of the last node you stopped. You will need this information when starting up the cluster.
5. Right-click the Spectrum Technology Platform icon in the Windows system tray and select **Stop Spectrum**.

   Warning: To prevent loss of data when starting the cluster, the first node you start must be the last node that was stopped, and that node must be a seed node.

Removing a Node from a Cluster

To remove a node from a cluster, stop the Spectrum Technology Platform server.

1. To stop the server, right-click the Spectrum Technology Platform icon in the Windows system tray (shown below) and select **Stop Spectrum**.
2. Stop the Spectrum Technology Platform server using the `../server/bin/server.stop` script.
3. Stop the node you want to remove:

   change the working directory to the Spectrum Technology Platform server's `bin` directory, source the setup file, then type the following command: `./server.stop` .

   On Windows, right-click the Spectrum Technology Platform icon in the system tray and select **Stop Spectrum**.

4. Open the file `server/conf/spectrum-container.properties` in a text editor and set `spectrum.cluster.enabled` to `false`.
5. On each of the other nodes in the cluster, open the `spectrum-container.properties` file and remove the node from the `spectrum.cluster.seeds` property.

For Spatial users: If you want to keep the node standalone and able to run outside the cluster, copy back the original `repository.xml` file and remove the following folders from the `/server/modules/spatial/jackrabbit` directory for each instance of Spectrum Technology Platform: repository, version, workspaces. Restart the server and import the repository content.

Managing a Cluster for Spectrum Spatial

Setting Up a Common Repository Database

You must configure Spatial to use a common repository database for the cluster. This ensures that named resources, geographic metadata and configuration settings are managed across the cluster.

The repository is installed with a set of named resources, geographic metadata and configuration files. To migrate these resources to the common database repository the resources need to be exported from the default internal repository database and reimported into the new shared repository database.

For bulk export and import of repository content, use the `limrepo import` and `limrepo export` commands in the Administration Utility. These commands give you the option of preserving permissions (see the Administration section of the *Spectrum Spatial Guide* for instructions.)

These steps describe how to set up your repository on a common database, either PostgreSQL, Oracle, or Microsoft SQL Server:

1. Export all repository resources to a local folder using the `limrepo export` command in the Administration Utility.

   For instructions, see the Administration section of the *Spectrum Spatial Guide*.

   The contents of the installed repository must be exported. This step only needs to be performed once, as the contents of the repository should be the same at this point for all instances of Spectrum Technology Platform.
2. Stop the Spectrum Technology Platform server on all nodes.

   For instructions, see **Stopping a Cluster** on page 37.
3. On all nodes of the Spectrum Technology Platform modify the configuration to specify the common database.

a) Copy the contents of `repository.databaseType.xml` to `repository.xml` located under the `server/modules/spatial/jackrabbit` folder where *databaseType* is the appropriate type for your database (`postgres`, `oracle`, or `mssql`).

b) In `repository.xml`:

- Modify the DataSource section with the server host name, port, database, user, and password.
- Modify the Cluster section to assign a distinct cluster ID, like Node1. Ensure unique IDs are assigned to every subsequent node in the cluster (for example, Node2, Node3).
- Save the changes to `repository.xml`.

c) Remove these folders from the `/server/modules/spatial/jackrabbit` folder: `repository`, `version`, `workspaces`.

4. If your database has previously contained any repository content, you must remove these tables to create a clean repository:

- default_binval
- default_bundle
- default_names
- default_refs
- rep_fsentry
- rep_global_revision
- rep_journal
- rep_local_revisions
- security_binval
- security_bundle
- security_names
- security_refs
- version_binval
- version_bundle
- version_names
- version_refs

If using Oracle, then also delete `version_seq_names_id`, `security_seq_names_id`, and `default_seq_names_id`.

5. On the seed node only, import the backed up repository content.

a) Start the Spectrum Technology Platform server.

For instructions, see **Starting a Cluster** on page 36.

b) Import the contents using the `limrepo import` command, pointing to the seed node.

6. Start the remaining nodes in the cluster.

For instructions, see **Starting a Cluster** on page 36.

Configuring Your System

Once the Spectrum Technology Platform is installed and you have configured a common repository, you need to configure your instance before you can replicate it to another virtual machine. If you are not using a virtual machine environment, you will need to perform these steps on each of your Spectrum Technology Platform installations.

Configure the Map File Share

To configure the map file share (a shared image folder) to Spectrum Technology Platform, you first need a shared map image directory.

> **Note:** To create a Linux map file share, see **Creating a Map Image File Share on Linux** on page 41.

> **Note:** To create a Windows map file share, see **Creating a Map Image File Share on Windows** on page 41.

Once a map image directory has been created, configure the map file share:

1. Modify the Mapping Service configuration by pointing to a shared image folder and load balance server. In the ImageCache change the Directory parameter to a common image directory, and change the `AccessBaseURL` parameter to the load balancer machine image URL.

   If you are using a virtual machine environment, remember this IP address, as you must set the load balancer VM to this IP address.

   For Linux installations:

```
<ImageCache>
<Directory>/<spatial server
root>/server/modules/spatial/images</Directory>
<AccessBaseURL>http://<loadbalance_IP_address>/rest/Spatial/
MappingService/internal/imageCache</AccessBaseURL>
 <FileExpire>30</FileExpire>
 <ScanInterval>30</ScanInterval>
</ImageCache>
```

   For Windows installations:

```
<ImageCache>
<Directory>\\server\Share\images</Directory>
<AccessBaseURL>http://<loadbalance_IP_address>/rest/Spatial/MappingService/
internal/imageCache
</AccessBaseURL>
 <FileExpire>30</FileExpire>
```

```
 <ScanInterval>30</ScanInterval>
</ImageCache>
```

2. For Linux installations, you must set up a symbolic link to enable map images to go to the shared file system.

   Create an `images` subfolder in the mounted share folder, for example, `/mnt/<linux mount>/images`

```
cd /<spatial server root>/server/modules/spatial
rm –Rf images
ln -s /mnt/<linux mount>/images ./images
```

Creating a Map Image File Share on Linux

The file share provides a folder to hold map images generated by Spectrum Spatial. Create a shared folder accessible to all Spectrum nodes. The file share is not required if maps are returned from the web services as Base64-encoded images.

To create a map image file share on Linux:

1. Mount a shared folder on each operating system hosting Spectrum. The commands below mount a drive on a Microsoft Windows Server or network drive supporting CIFS.

```
mkdir /mnt/<linux mount>
mount -t cifs //<windows host>/<windows share> /mnt/<linux mount>-o
username=<shareuser>,password=<sharepassword>,domain=<domain>
```

2. Set the image share to load at startup in `/etc/fstab`.

```
//<windows ip address for share>/share /path_to/mount cifs
username=server_user,password=secret,_netdev 0 0
```

Creating a Map Image File Share on Windows

The file share provides a folder to hold map images generated by Spectrum Spatial. Create a shared folder accessible to all Spectrum nodes. The file share is not required if maps are returned from the web services as Base64-encoded images.

To create a map image file share on Windows:

1. In Windows Explorer, select the image folder you want to share.
2. Right-click, then click **Share** or **Share with**.
3. Select the users who will be writing to the image folder. These users must have read/write privileges.

Modifying OGC Service Configurations for Clustering

To ensure clustering works when you have both a Spectrum Technology Platform cluster and a Spatial cluster, changes are required to the Open Geospatial Consortium (OGC) services configuration files using Spectrum Spatial Manager: From the WFS, WMS, and WMTS settings pages, change the online resource (service) URL to the IP address and port of the load balancer. See the *Spectrum Spatial Manager Guide* in the Utilities section of the *Spectrum Spatial Guide* for more information.

Configuring Ports for Multiple Spectrum Instances

If you have multiple Spectrum Technology Platform instances on a single machine, you must change the port numbers for each instance. Change all ports in `SpectrumDirectory/server/conf/spectrum-container.properties` to new port values that are not in use. The HTTP port reflects the port number entered in the installer.

Shared Spectrum Local Data

If you are using TAB file data on the file system, this data needs to be in a shared location accessible by all instances of Spectrum in the load balanced environment. It is also important to note that all named resources in the repository accessing data on the file system should point to this shared location.

Each VM or machine hosting Spectrum needs to have access to the mounted shared drive.

> **Note:** Using named resources that point to database tables do not require a shared drive, as the named resources in the repository do not access the data using a file path; rather they use a named connection to the data in the database.

6 - Using the Administration Utility

In this section

Getting Started with the Administration Utility

The Administration Utility provides command line access to administrative functions. You can execute the commands interactively or in scripts. Some administrative functions are not available in the Administration Utility. For these functions you can use the Spectrum Management Console.

Using a Script with the Administration Utility

The Administration Utility can execute a series of commands from a script file. This is useful if you want to automate or standardize administrative actions through the use of a script instead of manually executing commands through the Administration Utility or by using Spectrum Management Console.

1. Using a text editor, create a script file. A script file contains the commands that you want to execute.

   To add a command to a script file, type the command and the necessary parameters as you would if you were entering the command at the command prompt. Enter one command for each line.

   To insert comments into a script file, use the following notation:

`/*`	Indicates the start of a block comment.
`*/`	Indicates the end of a block comment.
`//`	Indicates an inline comment. Use at the start of a line only.
`;`	Indicates an inline comment. Use at the start of a line only.

2. Save the script either on the computer where you run the Administration Utility or in a location that is accessible from the computer where you run the Administration Utility. You can use any file name and extension you choose. The recommend file extension is `.cli`.
3. To execute the script, do one of the following:

Option	Description
To execute the script at the command line	Specify the following at the command line or in a batch or shell script: `cli.cmd --cmdfile` *`ScriptFile`*

Option	Description
To execute the script from the Administration Utility	Open the Administration Utility and connect to the Spectrum Technology Platform server using the `connect` command. Then, use the `script` command to execute the script. For more information on this command, see **system_script.dita**.

Example: Moving Dataflows from Staging to Production

You have three dataflows: Deduplication, AddressValidation, and DrivingDirections. You have a staging server where you make changes to these dataflows and test them, and a production environment where the dataflows are made available for execution. You want to have a consistent and automated way to move these dataflows from your staging server to your production server so you decide to use an Administration Utility script to accomplish this. The script might look like this:

```
// Connect to the staging server
connect --h stagingserver:8080 --u allan12 --p something123

// Export from staging
dataflow export --d "Deduplication" --e true --o exported
dataflow export --d "AddressValidation" --e true --o exported
dataflow export --d "DrivingDirections" --e true --o exported

// Close connection to the staging server
close

// Connect to the production server
connect --h productionserver:8080 --u allan12 --p something123

// Import to production
dataflow import --f exported\Deduplication.df
dataflow import --f exported\AddressValidation.df
dataflow import --f exported\DrivingDirections.df

// Close the connection to the production server
close
```

Spectrum Spatial

limrepo export

> **Note:** For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `limrepo export` command exports named resources (such as named tables) from the Spectrum Spatial repository to a local file system. You must have Spectrum Spatial installed to use this command.

Resources are exported with their full repository paths in the target folder. For example, if you run `limrepo export` --s /Samples/NamedTables --o C:\export, the tool creates `C:\export\Samples\NamedTables\WorldTable`, and so on for each named table under the NamedTables folder or directory.

> **Note:** The `limrepo export` command will always recursively export all folders, including empty ones.

Usage

`limrepo export --s` *SourceRepositoryPath* `--o` *OutputFilePath*

> **Note:** To see a list of parameters, type `help limrepo export`.

Required	Argument	Description
Yes	`--s` or `source`	Specifies the path to the resource or a folder to be exported.
Yes	`--o` or `output`	Specifies the path to a folder on the local file system where you want to export. This can be a new folder or an existing folder; however, an existing folder must be empty otherwise the export will fail.
No	`--q` or `--quiet`	Disables the display of the resources copied during the export; that is, operates in quiet mode. If the flag is specified, the default is true. If the flag is not specified, the default is false.

Required	Argument	Description
No	`--f` or `--fullpaths`	Prints the full source and output paths. If the flag is specified, the default is true. If the flag is not specified, the default is false.
No	`--r` or `--recursive`	Recursively exports subfolders (children of the specified source). **true** All files under the specified folder location and files under all subfolders export. This is the default setting if the flag is not specified or if the flag is specified without a value. **false** Only those files under the specified folder location export (subfolders do not export). Specifying false increases the possibility that the export will contain named resources with references to resources that have not been exported. This flag should be used with extreme caution and by those who understand all the relationships of their repository.
No	`--c` or `--continueonerror`	Continues with the export if an error occurs. If the flag is specified, the default is true. If the flag is not specified, the default is false.
No	`--a` or `--acl`	Preserves existing permissions for the exported resources in the export folder on the local file system. An access control list (ACL) indicates the operations each user or role can perform on a named resource, such as create, view, edit, or delete. If the flag is specified, the default value is true. If the flag is not specified, the default value is false.

Example

This example exports the named resources in the repository's \Samples folder to C:\myrepository\samples on your local file system.

```
limrepo export --s /Samples --o C:\myrepository\samples
```

limrepo import

> **Note:** For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `limrepo import` command imports named resources (such as named tables) from a local file system into the Spectrum Spatial repository. You must have Spectrum Spatial installed to use this command.

When importing, you must point to the same folder or directory you exported to previously. For example, if you run `limrepo export` --s /Samples/NamedTables --o C:\export, the tool creates `C:\export\Samples\NamedTables\WorldTable`, for each named table under the NamedTables folder or directory. Resources are exported with their full repository paths in the target folder. Running `limrepo import` --s C:\export then imports WorldTable back to `/Samples/NamedTables/WorldTable`.

> **Note:** The `limrepo import` command will always recursively import all folders, including empty ones.

After performing an import, in many cases, you will need to adjust the named connections to point to their new path using Spectrum Spatial Manager. For example, if your Native TAB files were installed on `C:\myfiles` in your test instance and the same files are installed on `E:\ApplicationData\Spectrum\Spatial\Q3` then that connection would have to be corrected in Spectrum Spatial Manager after import. See the Utilities section of the *Spectrum Spatial Guide* for instructions on using Spectrum Spatial Manager to edit a named connection.

> **Note:** If you are using `limrepo import` to restore service configuration files that you exported from a pre-12.0 version of Spectrum Technology Platform, the files will automatically be modified to be compliant with version 12.0 and later (for example, the repository URLs will be removed).

Usage

`limrepo import --s` *SourceFilePath*

> **Note:** To see a list of parameters, type `help limrepo import`.

Required	Argument	Description
Yes	`--s` or `source`	Specifies the path to the resource or a folder on the local file system that is to be imported. This must be the root folder of a previous export on the local file system.

Required	Argument	Description
No	`--q` or `--quiet`	Disables the display of the resources copied during the import; that is, operates in quiet mode. If the flag is specified, the default is true. If the flag is not specified, the default is false.
No	`--u` or `--update`	Specifies whether to overwrite existing resources if resources with the same name are already on the server. **true** If there is a resource on the server with the same name as a resource you are importing, the resource on the server will be overwritten. This is the default setting if the flag is not specified or if the flag is specified without a value. **false** If there is a resource on the server with the same name as a resource you are importing, the resource will not be imported.
No	`--f` or `--fullpaths`	Prints the full source and output paths. If the flag is specified, the default is true. If the flag is not specified, the default is false.
No	`--c` or `--continueonerror`	Continues with the import if an error occurs. If the flag is specified, the default is true. If the flag is not specified, the default is false.
No	`--a` or `--acl`	Preserves any previously exported permissions and merges them with existing permissions when importing resources. An access control list (ACL) indicates the operations each user or role can perform on a named resource, such as create, view, modify, or delete. For example, a user has read and write permissions on a resource when exporting. If the user only has read permissions on the resource when importing, write permission will be granted again after the import finishes successfully. Conflicting permissions cannot be merged and will be ignored. ACL entries for users and roles that do not exist in the target repository are also ignored.

Required	Argument	Description
		If the flag is specified, the default value is true. If the flag is not specified, the default value is false. **Tip:** When using this flag, the user on the server you exported from should also exist on the server to which you are importing. For example, you have "testuser" with access control settings and export the resources with ACL from one server, then import those named resources to another server that does not have "testuser". In this case, named resources will be uploaded but not the ACL.

Example

This example imports the named resources from C:\myrepository\samples on your local file system.

```
limrepo import --s C:\myrepository\samples
```

limrepo mwsimport

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `limrepo mwsimport` command in the Spectrum Technology Platform Administration Utility allows you to provision a map from a MapInfo Workspace (MWS) file that has been created either by MapInfo Pro or the MapXtreme Workspace Manager into the Spectrum Spatial repository. The import will create the named map and all its dependent resources (layers, tables and connections). The connection is named by appending 'Connection' to the map name. The named tables and named layers are created in subfolders (NamedTables and NamedLayers, respectively).

You must have Spectrum Spatial installed to use this command.

Usage

`limrepo mwsimport --s` *MWSFilePath* `--o` *Output* `--p` *ServerPath*

Note: To see a list of parameters, type `help limrepo mwsimport`.

Required	Argument	Description
Yes	`--s` or `source`	Specifies the path to an MWS file on the local file system that is to be imported.

Required	Argument	Description
Yes	`--o` or `output`	Specifies the path to the named map on the repository. All resources will be created within the same folder as the named map.
Yes	`--p` or `path`	Specifies the file path to the location of the data on the server. This path is used to create a named connection which is then referenced by all the named tables that are created. These tables will use file paths relative to that named connection.
No	`--l` or `local`	Specifies the file path to the location of the data on the local file system, if the MWS contains file paths that do not exist on the server file system. Any occurrences of the specified value in the MWS file will be substituted with the specified server path. If you have partial paths in the MWS file, this is not required; this is usually the case with anything created from MapXtreme.

Example

This example imports an MWS file on the D: drive (where the data on the server exists at C:\mydata) and provisions the named resources into /Europe/Countries in the repository.

```
limrepo mwsimport --s D:\europe.mws --o /Europe/Countries --p
C:\mydata
```

Result

The following named resources are created:

/Europe/Countries/Europe *(named map)*
/Europe/Countries/EuropeConnection *(named connection)*
/Europe/Countries/NamedTables/austria *(named table)*
/Europe/Countries/NamedTables/belgium *(named table)*
.
/Europe/Countries/NamedLayers/austria *(named layer)*
/Europe/Countries/NamedLayers/belgium *(named layer)*
..

Routing

ermdb list

> **Note:** For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `ermdb list` command retrieves a list of all the existing routing database resource on the server. You must have Spectrum Spatial installed to use this command.

Usage

```
ermdb list
```

> **Example**
>
> This example returns all the database resources on the server.
>
> ```
> ermdb list
> ```

ermdb get

> **Note:** For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `ermdb get` command allows you to return information on the routing databases configured on the server. Information returned is the name of the database, location of the database on the file system (path), and the pool size configured for the database. You must have Spectrum Spatial installed to use this command.

Usage

`ermdb get --name` *database_name*

> **Note:** To see a list of parameters, type `help ermdb get`.

Required	Argument	Description
Yes	`--name` or `--n` *database_name*	Specifies the name of the database resource to return information. The name must be a unique name on the server. For a list of existing routing database resources, use the `ermdb list` command.

Example

This example returns the information for the database resources US from the server.

```
ermdb get --name US
```

ermdb add

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `ermdb add` command creates a new routing database resource on the server. You must have the Spectrum Spatial installed to use this command.

Note: The `ermdb add` command requires a unique name be used for each of the databases being added.

Usage

`ermdb add --name` *database_name* `--poolsize` *pool_size* `--path` *database_path* `--mn` *minimum_memory_size* `--mx` *maximum_memory_size*

Note: To see a list of parameters, type `help ermdb add`.

Required	Argument	Description
Yes	`--name` or `--n` *database_name*	Specifies the name of the database resource to be added. The name must be a unique name on the server. For a list of existing routing database resources, use the `ermdb list` command.
No	`--poolsize` or `--s` *pool_size*	Indicates the maximum number of concurrent requests the database should handle. The default if not specified is 4. The accepted range for concurrent requests is any integer between 1 and 128.

Required	Argument	Description
Yes	`--path` *database_path*	Specifies the location of the routing database on the file server.
No	`--mn` or `--minMem` *minimum_memory_size*	Defines the minimum amount of memory allocated for this database. This value must be less than or equal to the `--mx` setting.
No	`--mx` or `--maxMem` *maximum_memory_size*	Defines the maximum amount of memory allocated for this database. This value must be greater than zero, but cannot exceed 65536 MB.

Example

This example adds the database resources US from `E:/ERM-US/2019.09/driving/south` into the server.

```
ermdb add --name US --poolsize 10
--path E:/ERM-US/2019.09/driving/south --mn 1200 --mx 65536
```

ermdb delete

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `ermdb delete` command removes an existing routing database resource from the server. You must have the Spectrum Spatial installed to use this command.

Usage

`ermdb delete --name` *database_name*

Note: To see a list of parameters, type `help ermdb delete`.

Required	Argument	Description
Yes	`--name` or `--n` *database_name*	Specifies the name of the database resource to be deleted. For a list of existing routing database resources, use the `ermdb list` command.

Example

This example removes the database resources US from the server.

```
ermdb delete --name US
```

ermdb modify

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `ermdb modify` command changes an existing routing database resource on the server. You must have Spectrum Spatial installed to use this command.

Usage

`ermdb modify --name` *database_name* `--poolsize` *pool_size* `--path` *database_path* `--o` *override* `--replytimeout` *value* `--vmargs` *java_argument*

Note: To see a list of parameters, type `help ermdb modify`.

Required	Argument	Description
Yes	`--name` or `--n` *database_name*	Specifies the name of the database resource to be modified. For a list of existing routing database resources, use the `ermdb list` command.
No	`--poolsize` or `--s` *pool_size*	Indicates the maximum number of concurrent requests the database should handle. The accepted range for concurrent requests is any integer between 1 and 128. You must specify either a new pool size or a new database path.
No	`--path` *database_path*	Specifies the new location of the routing database on the file server. You must specify either a new pool size or a new database path.
No	`--o` *override*	Overrides the default database settings in the Spectrum Management Console. The override is either *true* or *false*. The default value is *false*.
No	`--replytimeout` *value*	Sets a timeout message in the response based on the time value that you set. The value must be an integer and represents minutes. The default value is *0* minutes. Use this override option with `--o` *true*.

Required	Argument	Description
No	`--vmargs` *java_argument*	Sets the Java Virtual Machine Arguments (VMArgs) value to add a database to the Spectrum server. Use this override option with `--o` *true*.

Examples

This example modifies both the pool size and the database path for a new vintage.

```
ermdb modify --name US --poolsize 20 --path
E:/ERM-US/2015.03/driving/south
```

This example uses the `--o` *override* option to override settings made in the Spectrum Management Console for a saved database (in the **Override** section for a saved database).

```
ermdb modify --name US --poolsize 10 --path
D:/USA_092018/US_Driving/northeast --mn 2096 --mx 4096 --o
true
```

This example sets the `--o` *override* option to *true* and sets the `--replytimeout` value of the response to 2 minutes. The `--replytimeout` option is an override option, so use it with `--o` *true*.

```
- ermdb modify --name US --poolsize 10 --path ermdb modify
--name US --poolsize 10 --path
D:/USA_092018/US_Driving/northeast --mn 2096 --mx 4096 --o
true --replytimeout 2
```

This example uses the `--vmargs` option to set the path of the database that is present in the local system. The `--vmargs` option is an override option, so use it with `--o` *true*.

```
rmdb modify --name US --poolsize 10 --path
D:/USA_092018/US_Driving/northeast --mn 2096 --mx 3096 --o
true --replytimeout 2 -vmargs -Xmx4096
```

ermdb import

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `ermdb import` command allows you to import a file consisting of routing database configurations and creates the database resources on the server. You can create the import file, use the file created by the `ermdb template` command, or use the file created by the `ermdb export` command. You must have Spectrum Spatial installed to use this command.

The import file format is as follows:

```
[
{
    "product": "Spatial",
    "module": "routing",
    "name": "US",
    "maxActive": 4,
    "properties":
        {
            "DatasetPaths": "E:/ERM-US/2014.09/driving/northeast"
        }
    }
]
```

Where `product` and `module` must be Spatial and routing, `name` is the name of the database, `maxActive` is the maximum number of concurrent requests you want this database to handle (or the pool size), and `DatasetPaths` is the path to the data sets for the database resource.

You can add multiple databases in an import file (duplicate the example above), and add multiple datasets for each database resource separating them using semi colons.

> **Note:** If you want to specify UTF-8 characters in import file, you must add the JVM parameter file.encoding to the value UTF-8 in the startup of the CLI command prompt; for example: `-Dfile.encoding=UTF-8`

Usage

`ermdb import --file` ***file_name***

> **Note:** To see a list of parameters, type `help ermdb import`.

Required	Argument	Description
YES	`--file` or `--f` *file_name*	Specifies the directory and name of the import file.

Example

This example imports two databases US1 and US2 each consisting of multiple datasets.

```
ermdb import --file E:/ERM-US/export/ermDbResource.txt
```

The input file is defined as:

```
[
{
        "product": "Spatial",
        "module": "routing",
        "name": "US1",
```

```
        "maxActive": 4,
        "properties":
        {
            "DatasetPaths":
"E:/ERM-US/2014.09/driving/northeast;E:/ERM-US/2014.09/driving/south"
        }
    },
{
        "product": "Spatial",
        "module": "routing",
        "name": "US2",
        "maxActive": 4,
        "properties":
        {
            "DatasetPaths":
"E:/ERM-US/2014.09/driving/northeast;E:/ERM-US/2014.09/driving/central"

        }
    }
]
```

ermdb export

> **Note:** For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `ermdb export` command allows you to export the routing databases configured on the server to a file. This file can then be used to import into another instance using the `ermdb import` command, either as a backup, or for migration from one instance to another. You must have Spectrum Spatial installed to use this command.

> **Note:** The `ermdb export` command will always create an export file name `ermDbResource.txt`

Usage

`ermdb export --directory` *directory_name*

> **Note:** To see a list of parameters, type `help ermdb export`.

Required	Argument	Description
No	`--directory` or `--o` *directory_name*	Specifies the name of the directory on the file system where to export the database file. The export command will always create an export file

Required	Argument	Description
		name `ermDbResource.txt`. If this parameter is not specified, the export file will be created in the directory where the export command is run.

Example

This example creates an export database file in the `E:/ERM-US/export` directory.

```
ermdb export --directory E:/ERM-US/export
```

Log File Contents

Exporting a database that was added using the Spectrum Management Console generates a log file. The logfile is named `ermDbResource.txt` and it is created in the same folder where the CLI export command is run. If the exported database had CLI **ermdb modify** options applied to it, then the log file includes the settings for the CLI options. The following is an example from a log file that included CLI option settings information.

```
[{"product":"Spatial",
"module":"routing",
"minimumMemory":2096,
"maximumMemory":4096,
"name":"US",
"override":true,
"replyTimeout":0,
"vmargs":"",
"properties":{"DatasetPaths":"D:/USA_092018/US_Driving/northeast"},
"maxActive":10}]
```

erm getpointdata

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `erm getpointdata` command returns segments information for a point. The closest segments are returned to the specified point. Types of information returned are; segment ID, road type, length, speed, direction, time, road name. You must have Spectrum Spatial installed to use this command.

Usage

`erm getpointdata --datasource` *db_resource* `--point` *"x,y,coordsys"*

Note: To see a list of parameters, type `help erm getpointdata`.

Required	Argument	Description
Yes	`--datasource` *db_resource*	Specifies the name of the database resource to return data. For a list of existing routing database resources, use the `ermdb list` command.
Yes	`--point` *"x,y,coordsys"*	Indicates the point to return the closest segment information. The point is specified in the format *"x,y,coordsys"*, where *coordsys* is the coordinate system of the point.

Example

This example returns the closest segment data to the specified point from the US_NE database resources configured on the server.

```
erm getpointdata --datasource US_NE --point "-72,40,epsg:4326"
```

erm getsegmentdata

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `erm getsegmentdata` command returns segments information for a given segment ID. Types of information returned are; segment ID, road type, length, speed, direction, time, road name. You must have the Spectrum Spatial installed to use this command.

Usage

`erm getsegmentdata --datasource` *db_resource* `--segmentid` *"segment_id"*

Note: To see a list of parameters, type `help erm getsegmentdata`.

Required	Argument	Description
Yes	`--datasource` *db_resource*	Specifies the name of the database resource to return data. For a list of existing routing database resources, use the `ermdb list` command.
Yes	`--segmentid` *"segment_id"*	Indicates the segment to return the information. The segment is specified in the format specified in the data. For example, *"7e3396fc:6e5251"*.

Example

This example returns data for the specified segment from the US_NE database resources configured on the server.

```
erm getsegmentdata --datasource US_NE --segmentid
"7e3396fc:6e5251"
```

erm createpointupdate

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `erm createpointupdate` command overrides the routing data of the closest segment for a given point. This command allows you to set or change the speed, or exclude a section of the route. You must have Spectrum Spatial installed to use this command.

Note: The type of persistent update is valid only for the specified data resource and may not be valid after a data update.

Usage

`erm createpointupdate --datasource` *db_resource* `--point` *"x,y,coordsys"* `--exclude --velocity` *velocity_value* `--velocityunit` *velocity_unit* `--velocityadjustment` *velocity_adjustment_value* `--velocitypercentage` *velocity_percentage_value*

Note: To see a list of parameters, type `help erm createpointupdate`.

Required	Argument	Description
Yes	`--datasource` *db_resource*	Specifies the name of the database resource to override the data. For a list of existing routing database resources, use the `ermdb list` command.
Yes	`--point` *"x,y,coordsys"*	Indicates the point to override the closest segment information. The point is specified in the format *"x,y,coordsys"*, where *coordsys* is the coordinate system of the point.
No	`--exclude`	Excludes the specified point from all route calculations when set as `true`. Having this parameter in the command specifies whether to exclude the point. To avoid the exclusion, add `false` after --exclude.

Required	Argument	Description
No	`--velocity` *velocity_value*	Defines a speed update where you specify the new speed of the point by specifying the new velocity. The default unit is mph(miles per hour) unless you specify the `velocityunit` parameter.
No	`--velocityunit` *velocity_unit*	Defines a unit of speed for the `velocity` or `velocityadjustment` overrides. The default is mph(miles per hour). For speed updates, the velocity unit can have one of the following values: kph (kilometers per hour), mps(meters per second), or mph (miles per hour).
No	`--velocityadjustment` *velocity_adjustment_value*	Defines a speed update where you define a change in the speed of the point by specifying the change in velocity (unit and value). Speed values can be increased (positive value) or decreased(negative value). The default unit is mph(miles per hour) unless you specify the `velocityunit` parameter.
No	`--velocitypercentage` *velocity_percentage_value*	Defines a speed update where you define an increase in the speed of the point by specifying a percentage to increase(positive value) or decrease(negative value) the speed.

Examples

This example overrides the speed of the point to 15 mph, from the US_NE database resources configured on the server.

```
erm createpointupdate --datasource US_NE --point
"-72,40,epsg:4326" --velocity 15 --velocityunit mph
```

This example excludes the specified point from the US_NE database resources configured on the server.

```
erm createpointupdate --datasource US_NE --point
"-72,40,epsg:4326" --exclude true
```

This example overrides the speed of the point by increasing the speed by 45 kph, from the US_NE database resources configured on the server.

```
erm createpointupdate --datasource US_NE --point
"-72,40,epsg:4326" --velocityadjustment 45 --velocityunit kph
```

This example overrides the speed of the point by decreasing the speed by 60 percent, from the US_NE database resources configured on the server.

```
erm createpointupdate --datasource US_NE --point
"-72,40,epsg:4326" --velocitypercentage -60
```

erm resetpointupdate

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `erm resetpointupdate` command returns any overrides to the original state of the data. You must have Spectrum Spatial installed to use this command.

Usage

`erm resetpointupdate --datasource` *db_resource* `--point` *"x,y,coordsys"* `--resettype` *reset_type*

Note: To see a list of parameters, type `help erm resetpointupdate`.

Required	Argument	Description
Yes	`--datasource` *db_resource*	Specifies the name of the database resource that has the overrides. For a list of existing routing database resources, use the `ermdb list` command.
Yes	`--point` *"x,y,coordsys"*	Indicates the point where the existing overrides are located. The point is specified in the format *"x,y,coordsys"*, where *coordsys* is the coordinate system of the point.
Yes	`--resettype` *reset_type*	The type of override to remove (undo). **speed** Removes a speed update. **exclude** Removes an exclude update.

Example

This example resets an existing exclude override for the given point, from the US_NE database resources configured on the server.

```
erm resetpointupdate --datasource US_NE --point
"-72,40,epsg:4326" --resettype exclude
```

erm createsegmentupdate

> **Note:** For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `erm createsegmentupdate` command overrides the routing data of the specified segment. This command allows you to set or change the speed, exclude a section of the route, or change the road type. You must have Spectrum Spatial installed to use this command.

> **Note:** The type of persistent update is valid only for the specified data resource and may not be valid after a data update.

Usage

`erm createsegmentupdate --datasource` *db_resource* `--segmentid` *"segment_id"* `--exclude --velocity` *velocity_value* `--velocityunit` *velocity_unit* `--velocityadjustment` *velocity_adjustment_value* `--velocitypercentage` *velocity_percentage_value* `--roadtype` *road_type*

> **Note:** To see a list of parameters, type `help erm createsegmentupdate`.

Required	Argument	Description
Yes	`--datasource` *db_resource*	Specifies the name of the database resource to override the data. For a list of existing routing database resources, use the `ermdb list` command.
Yes	`--segmentid` *"segment_id"*	Indicates the segment to override. The segment is specified in the format specified in the data. For example, *"7e3396fc:6e5251"*.
No	`--exclude`	Excludes the specified segment from all route calculations when set to `true`. Having this parameter in the command specifies whether to exclude the segment. To avoid the exclusion, add `false` after --exclude.
No	`--velocity` *velocity_value*	Defines a speed update where you specify the new speed of the segment by specifying the new velocity. The default unit is mph(miles per hour) unless you specify the `velocityunit` parameter.
No	`--velocityunit` *velocity_unit*	Defines a unit of speed for the `velocity` or `velocityadjustment` overrides. The value is

Required	Argument	Description
		mph(miles per hour). For speed updates, the velocity unit can have one of the following values: kph (kilometers per hour), mps(meters per second), or mph (miles per hour).
No	`--velocityadjustment` *velocity_adjustment_value*	Defines a speed update where you define a change in the speed of the segment by specifying the change in velocity (unit and value). Speed values can be increased (positive value) or decreased(negative value). The default unit is mph(miles per hour) unless you specify the `velocityunit` parameter.
No	`--velocitypercentage` *velocity_percentage_value*	Defines a speed update where you define an increase in the speed of the segment by specifying a percentage to increase(positive value) or decrease(negative value) the speed.
No	`--roadtype` *road_type*	Defines the new road type for the segment.

Examples

This example overrides the speed of the segment to 15 mph, from the US_NE database resources configured on the server.

```
erm createsegmentupdate --datasource US_NE --segmentid
"7e3396fc:6e5251" --velocity 15 --velocityunit mph
```

This example excludes the specified segment from the US_NE database resources configured on the server.

```
erm createsegmentupdate --datasource US_NE --segmentid
"7e3396fc:6e5251" --exclude true
```

This example overrides the speed of the segment by increasing the speed by 45 kph, from the US_NE database resources configured on the server.

```
erm createsegmentupdate --datasource US_NE --segmentid
"7e3396fc:6e5251" --velocityadjustment 45 --velocityunit kph
```

This example overrides the speed of the segment by decreasing the speed by 60 percent, from the US_NE database resources configured on the server.

```
erm createsegmentupdate --datasource US_NE --segmentid
"7e3396fc:6e5251" --velocitypercentage -60
```

This example overrides the road type of the segment to ferry, from the US_NE database resources configured on the server.

```
erm createsegmentupdate --datasource US_NE --segmentid
"7e3396fc:6e5251" --roadtype ferry
```

erm resetsegmentupdate

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `erm resetsegmentupdate` command returns any overrides to the original state of the data. You must have the Spectrum Spatial installed to use this command.

Usage

`erm resetsegmentupdate --datasource` ***db_resource*** `--segmentid` ***"segment_id"*** `--resettype` ***reset_type***

Note: To see a list of parameters, type `help erm resetsegmentupdate`.

Required	Argument	Description
Yes	`--datasource` ***db_resource***	Specifies the name of the database resource that has the overrides. For a list of existing routing database resources, use the `ermdb list` command.
Yes	`--segment` ***"segment_id"***	Indicates the segment where the existing overrides are located. The segment is specified in the format specified in the data. For example, *"7e3396fc:6e5251"*.
Yes	`--resettype` ***reset_type***	The type of override to remove (undo). **speed** Removes a speed update. **exclude** Removes an exclude update. **roadtype** Removes a road type update.

Example

This example resets an existing road type override for the given segment, from the US_NE database resources configured on the server.

```
erm resetsegmentupdate --datasource US --segmentid
"7e3396fc:6e5251" --resettype roadtype
```

erm getsegmentupdates

> **Note:** For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `erm getsegmentupdates` command returns a list of overrides in the routing data for the specified segments. You must have Spectrum Spatial installed to use this command.

> **Note:** `segmentids` is an optional parameter. If no segment ids are specified, then overrides for all available segments are returned.

Usage

`erm getsegmentupdates --datasource` *db_resource* `--segmentids` *"segment_ids"* `--velocityunit` *velocityunit*

> **Note:** To see a list of parameters, type `help erm getsegmentupdates`.

Required	Argument	Description
Yes	`--datasource` *db_resource*	Specifies the name of the database resource that has overrides. For a list of existing routing database resources, use the `ermdb list` command.
No	`--segmentids` *"segment_ids"*	A comma separated list of segment ids to return override information. Segments are specified in the format specified in the data. For example, *"7e3396fc:6e5251"*.
No	`--velocityunit` *velocityunit*	Specifies the velocity unit to appear in the response (mph - miles per hour, kph - kilometers per hour, mtps - meters per second, and mtpm - meters per minute). The default is mph.

Example

This example returns the overrides for a segment, from the US_NE database resources configured on the server.

```
erm getsegmentupdates --datasource US_NE --segmentids
"7e3396fc:6e5251" --velocityunit kph
```

erm createroadtypeupdate

> **Note:** For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `erm createroadtypeupdate` command overrides the routing data of the specified road type. This command allows you to set or change the speed of the route for the particular road type. You must have Spectrum Spatial installed to use this command.

> **Note:** The type of persistent update is valid only for the specified data resource and may not be valid after a data update.

Usage

`erm createroadtypeupdate --datasource` *db_resource* `--roadtype` *"road_type"* `--velocity` *velocity_value* `--velocityunit` *velocity_unit* `--velocityadjustment` *velocity_adjustment_value* `--velocitypercentage` *velocity_percentage_value* `--roadtype` *road_type*

> **Note:** To see a list of parameters, type `help erm createroadtypeupdate`.

Required	Argument	Description
Yes	`--datasource` *db_resource*	Specifies the name of the database resource to override the data. For a list of existing routing database resources, use the `ermdb list` command.
Yes	`--roadtype` *"road_type"*	Indicates the road type to override: • access way • back road • connector • ferry • footpath • limited access dense urban • limited access rural • limited access suburban • limited access urban • local road dense urban • local road rural • local road suburban • local road urban • major local road dense urban

Required	Argument	Description
		• major local road rural • major local road suburban • major local road urban • major road dense urban • major road rural • major road suburban • major road urban • minor local road dense Urban • minor local road rural • minor local road suburban • minor local road urban • normal road dense urban • normal road rural • normal road rural • normal road urban • primary highway dense urban • primary highway rural • primary highway suburban • primary highway urban • ramp dense urban • ramp limited access • ramp major road • ramp primary highway • ramp rural • ramp secondary highway • ramp urban • ramp suburban • secondary highway dense urban • secondary highway rural • secondary highway suburban • secondary highway urban
No	`--velocity` *velocity_value*	Defines a speed update where you specify the new speed of the road type by specifying the new velocity. The default unit is mph(miles per hour) unless you specify the `velocityunit` parameter.
No	`--velocityunit` *velocity_unit*	Defines a unit of speed for the `velocity` or `velocityadjustment` overrides. The default is mph(miles per hour). For speed updates, the velocity unit can have one of the following values:

Required	Argument	Description
		kph (kilometers per hour), mps(meters per second), or mph (miles per hour).
No	`--velocityadjustment` *velocity_adjustment_value*	Defines a speed update where you define a change in the speed of the road type by specifying the change in velocity (unit and value). Speed values can be increased (positive value) or decreased(negative value). The default unit is mph(miles per hour) unless you specify the `velocityunit` parameter.
No	`--velocitypercentage` *velocity_percentage_value*	Defines a speed update where you define an increase in the speed of the road type by specifying a percentage to increase(positive value) or decrease(negative value) the speed.

Examples

This example overrides the speed of a road type to 25 kph, from the US_NE database resources configured on the server.

```
erm createroadtypeupdate --datasource US_NE --roadtype "normal road suburban" --velocity 25 --velocityunit kph
```

This example increases the speed of the specified road type by 50 kph, from the US_NE database resources configured on the server.

```
erm createroadtypeupdate --datasource US_NE --roadtype "normal road suburban" --velocityadjustment 50 --velocityunit mph
```

This example overrides the speed of the road type by decreasing the speed by 65 percent, from the US_NE database resources configured on the server.

```
erm createroadtypeupdate --datasource US_NE --roadtype "normal road suburban" --velocitypercentage -65
```

erm resetroadtypeupdate

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `erm resetroadtypeupdate` command returns any overrides to the original state of the data. You must have the Spectrum Spatial installed to use this command.

Usage

```
erm resetroadtypeupdate --datasource db_resource --roadtype "road_type"
```

Note: To see a list of parameters, type `help erm resetroadtypeupdate`.

Required	Argument	Description
Yes	`--datasource` *db_resource*	Specifies the name of the database resource that has the overrides. For a list of existing routing database resources, use the `ermdb list` command.
Yes	`--roadtype` *"road_type"*	Indicates the road type that has the existing overrides. For a list of road types, see **erm createroadtypeupdate** on page 68.

Example

This example resets the "normal road suburban" road type override, from the US_NE database resources configured on the server.

```
erm resetroadtypeupdate --datasource US_NE --roadtype "normal road suburban"
```

erm getroadtypeupdates

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `erm getroadtypeupdates` command returns a list of overrides in the routing data for the specified road types. You must have Spectrum Spatial installed to use this command.

Note: `roadtypes` is an optional parameter. If no road types are specified, then overrides for all available road types are returned.

Usage

```
erm getroadtypeupdates --datasource db_resource --roadtypes "road_types"
--velocityunit velocityunit
```

Note: To see a list of parameters, type `help erm getroadtypeupdates`.

Required	Argument	Description
Yes	`--datasource` *db_resource*	Specifies the name of the database resource that has overrides. For a list of existing routing database resources, use the `ermdb list` command.
No	`--roadtypes` *"road_types"*	A comma separated list of road types to return override information. For a list of road types, see **erm createroadtypeupdate** on page 68.
No	`--velocityunit` *velocityunit*	Specifies the velocity unit to appear in the response (mph - miles per hour, kph - kilometers per hour, mtps - meters per second, and mtpm - meters per minute). The default is mph.

Example

This example returns the overrides for the "normal road urban" road type, from the US_NE database resources configured on the server.

```
erm getroadtypeupdates --datasource US_NE --roadtypes "normal road urban" --velocityunit kph
```

erm getallupdates

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `erm getallupdates` command returns a list of all overrides for a specified routing database resource. You must have Spectrum Spatial installed to use this command.

Usage

`erm getallupdates --datasource` *db_resource "segment_ids"* `--velocityunit` *velocityunit*

Note: To see a list of parameters, type `help erm getallupdates`.

Required	Argument	Description
Yes	`--datasource` *db_resource*	Specifies the name of the database resource that has the overrides. For a list of existing routing database resources, use the `ermdb list` command.
No	`--velocityunit` *velocityunit*	Specifies the velocity unit to appear in the response (mph - miles per hour, kph - kilometers per hour,

Required	Argument	Description
		mtps - meters per second, and mtpm - meters per minute). The default is mph.

Example

This example returns all the overrides from the US_NE database resources configured on the server.

```
erm getallupdates --datasource US_NE --velocityunit kph
```

erm resetallupdates

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 44.

The `erm resetallupdates` command returns all overrides to the original state of the data. You must have Spectrum Spatial installed to use this command.

Usage

`erm resetallupdates --datasource` *db_resource*

Note: To see a list of parameters, type `help erm resetallupdates`.

Required	Argument	Description
Yes	`--datasource` *db_resource*	Specifies the name of the database resource that has the overrides. For a list of existing routing database resources, use the `ermdb list` command.

Example

This example resets all overrides from the US_NE database resources configured on the server.

```
erm resetallupdates --datasource US_NE
```

7 - Routing

In this section

Specifying Default Service/Stage Options

Default options control the default behavior of each service or stage on your system. You can specify a default value for each option. The default option takes effect when a request does not explicitly define a value for a given option. These default options are also the settings used by default when you create a dataflow in Enterprise Designer using this service.

For information about the options, see the Stages and Resources and Data sections in the *Spectrum Spatial Guide* that apply to Spectrum Spatial.

> **Note:** Persistent Updates are not managed using the Management Console. To make persistent updates, use the spectrum command line functionality in the Administration Utility.
>
> **Note:** The Get Route Data service in the Management Console does not set default options, rather it is an interactive way to return routing data for segments. For more information on Get Route Data, see **Getting Route Data using Management Console** on page 77.

1. Open Management Console.
2. Go to the **Services** menu and select **Routing**.
3. Click the service you want to configure from the list on the left.
4. Set the options for the service. Most services have various types of options that appear on different tabs.
5. Click **Save**.

Previewing a Service/Stage

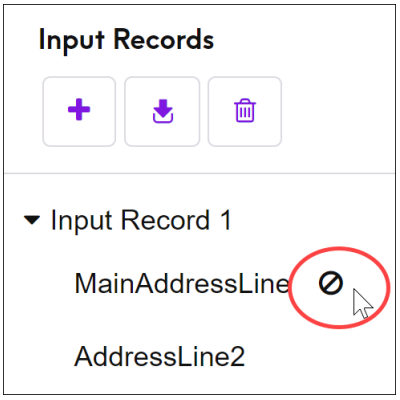
You can preview the results of a service in Management Console using the service's Preview tab. Preview can be useful in helping you decide what options to specify because you can immediately see the effect that different options have on the data returned by the service or stage.

1. Open Management Console.
2. Go to the **Services** menu and select the service you want to preview.
3. Click the **Preview** tab.
4. Enter the test data into each field.

   Here are some tips for using preview:

   - You do not have to enter data in every field. Leaving a field empty results in an empty string being used for preview.

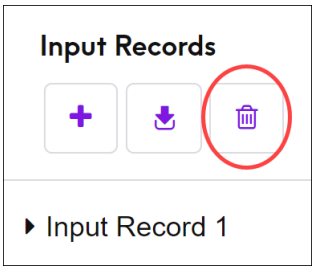
- If you want to preview the effect of passing a null value in a field, click the Disable icon next to the field:



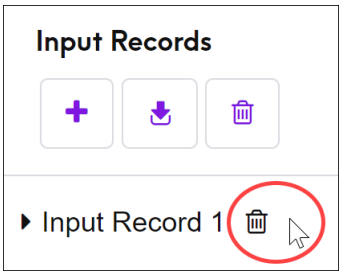
- You can preview multiple records at once. To add a record, click the Add button .
- You can import test data from a file. To import data, click the Import button . Select the **File name** and the **Field separator**. Note the following:
  - The first row in the file must be a header record. The field names in the header must match the field names required by the service.
  - The maximum number of records that can be imported is five.
  - If the file uses a space as the field separator, field values must be surrounded by quotes. Here is an example of a file that uses a space as the field separator:

```
AddressLine1 AddressLine2 City StateProvince PostalCode
"One Global View" "" "Troy" "NY" "12180"
"3001 Summer St" "" "Stamford" "CT" "06926"
"224 N Michigan Ave" "Suite 300" "Chicago" "IL" ""
```

- To delete all records, click the Delete button at the top of the preview area:



- To delete an individual record, hover over the input record name (for example, "Input Record 1") and click the Delete button next to the record name:



- If the service takes hierarchical input data:
  - To add child records, hover over the parent record and click the Add button.
  - To delete all children of a parent, hover over the parent record and click the Delete button.
  - To delete individual child records, hover over the child record and click the Delete button.

5. Click **Run Preview**.

   The service processes the input records and displays the results

6. Review your output data, making sure the results are what you intended to get from the service or stage. If necessary you can make changes to the option and click **Run Preview** again. (You do not need to input the data again.)

Getting Route Data using Management Console

Using the Management Console, you can preview and save segment information either from a closest point or segment ID. The GetRouteData service returns segment information for a point or segment ID. When a point is specified, the closest route segments are returned. When a segment ID is specified, the route data for that specified route segment is returned.

To preview and/or save route data:

1. Open Management Console.
2. Go to the **Services** menu and select **Routing**.
3. Select **Get Route Data** from the services list.
4. Select either Point Data or Segment Data from the **Input Type** field.
5. Select the routing database resource from the **Database** field.

   If you need to add a new routing database resource, see **Creating a Routing Database Resource**.

6. Enter the required information for the Input Type you selected.

   If you selected Point Data, enter the point coordinates and the coordinate system. If you selected Segment Data, enter the segment ID.

7. Click **Preview**.

   The route segment data is returned in the **Output Data** section. When there are more than one segments associated with the input, the multiple segments will be listed with Segment Details 1, Segment Details 2, etc.

8. Click either the **Save** button to save the routing data results as a text file, or the **Clear** button to remove the results from the Output Data.

8 - Troubleshooting Your System

In this section

Rebuilding a Corrupt Repository Index

Sometimes the repository can become corrupt if the server is shut down abruptly or the Java process is killed (manually or due to a power outage). As a result, you may be unable to get resources that were previously searchable, and there will be no errors or warnings in the logs. Once you verify that permission changes are not the cause, rebuild the index to fix this issue:

1. Shut down the server.
2. Delete the index directory at the following locations:
   - <Spectrum>\server\modules\spatial\jackrabbit\workspaces\default
   - <Spectrum>\server\modules\spatial\jackrabbit\workspaces\security
   - <Spectrum>\server\modules\spatial\jackrabbit\repository
3. Restart the server.
   Jackrabbit re-creates the index at the above locations while booting.

After rebuilding the index, the search works correctly again.

Inspecting a Non-Responsive Server

Spectrum runtime consists of multiple JVM processes. The primary process is the Spectrum server process. When the primary server process is not responding, Spectrum does not respond to requests. However, each remote component runs in a separate process. When a remote component process stops responding, only the corresponding functions do not work correctly. Other processes may continue to work. The following steps identify the process to check when there is no response to a request.

Identify which Process is not Responding

Spectrum Management Console: Connect to the Spectrum Management Console (`http://<server>:<port>/managementconsole`). If the Spectrum Management Console loads and functions as expected, then the main Spectrum server process is OK.

Spatial Services: On the Spectrum Technology Platform software page (`http://<server>:<port>/dcg`), select **Spectrum Spatial** from the menu. If the **Spectrum Spatial** service demo page loads and works as expected, then the Spectrum Spatial remote component process is OK.

- `http://<servername>:<port>/Spatial/MappingService/DemoPage.html`

- `http://<servername>:<port>/Spatial/FeatureService/DemoPage.html`

WebDAV API: Connect to the Repository WebDAV API (`http://<server>:<port>/RepositoryService/repository/default`) to ensure the repository is working as expected.

Enable JMX

After identifying the process that may have a problem, enable JMX monitoring on that process.

For the Spectrum server process, add the following system properties in `%Spectrum%/server/bin/wrapper/wrapper.conf` and restart the Spectrum server. For a Spectrum Spatial remote component, add the following system properties in the Spectrum Management Console by selecting from the main menu **Resources** and **Spatial**, and then selecting **Process arguments**.

```
-Djava.rmi.server.hostname=<hostname>
-Dcom.sun.management.jmxremote.port=9990
-Dcom.sun.management.jmxremote.rmi.port=9990
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.authenticate=false
```

Connect to the JMX Server

Launch JConsole (`%java%\bin\jconsole.exe`) from any workstation that has a compatible Java SDK installed on it, and that can access Spectrum directly on the network. In JConsole, select **Remote process** and use the connection string, `<server>:9990`. If an SSL (secure sockets layer) warning displays, select **insecure connection**.

Inspect the Process

1. Monitoring JVM memory usage

   When Java Virtual Machine (JVM) is running out of memory (OOM), the process is blocked by garbage collection (GC). Check the JVM memory usage on the **Memory** tab in JConsole. If the memory size causes the server to be non-responsive, refer to the instructions under **Increasing Heap Memory for the Spatial Remote Component** on page 82 to increase the heap size (memory).

2. Make a heap dump of the JVM process

   A memory leak may be one of the reasons that cause OOM. A JVM heap dump (file) can help identify memory leak issues. To find the process ID and generate a heap dump (file), log on to the server machine where Spectrum is running.

   ```
   ps aux | grep java | grep " -Dspatial" (or " -Dspectrum"  for the
   primary server process)
   jmap -dump:format=b,file=<file-path> <pid>
   ```

For example: `%java%/bin/jmap -dump:format=b,file=./heapdump.bin 12345`

3. Make a thread dump of the JVM process

   Thread deadlock may be one of the reasons that prevent the process from responding. A JVM thread dump (file) can help identify any deadlocks. Log on to the server machine where Spectrum is running to find the process ID and generate a thread dump (file).

```
ps aux | grep java | grep " -Dspatial"
jstack -F <pid> > <file-path>
```

   For example: `%java%/bin/jstack -F 12345 > ./threaddump.txt`

Increasing Heap Memory for the Spatial Remote Component

To increase the heap memory for Spectrum Spatial remote component that handles spatial service requests:

1. Open the Management Console.
2. From **Resources**, click **Spatial**.
3. Change the **Max memory** size to something larger than the default value of 2048MB (2GB).

   To increase the memory of the remote component to 4GB, change the value to 4096MB for example.

   > **Note:** Do not exceed the maximum memory available to your operating system and leave enough space for the operating system to do its work.

4. Click **Save**.

Changes to these properties take effect immediately.

precisely

2 Blue Hill Plaza, #1563
Pearl River, NY 10965
USA

www.precisely.com