precisely

Spectrum Technology Platform

Spectrum Administration Guide

Version 2022.1.0

Table of Contents

1 - Getting Started

Configuring a New System.....5
Accessing Spectrum Management Console.....6
Starting and Stopping the Server.....8
Installing the Client Tools.....9
Network Ports.....10
Configuring license and expiration notification....13

2 - Approval Flows

Create an approval flow for an entity.....16
Create an approval flow for an Exception type....17

3 - Security

Security Model.....20
Users.....21
Roles.....28
Access Control.....38
Security for Spatial.....41
Limiting Server Directory Access.....47
Using HTTPS.....49
Web Service Authentication.....55
Using LDAP or Active Directory for Authentication.....59
Implementing Spectrum Single Sign-on (SSO)....69
Encryption.....77

4 - Data Sources

Connections.....95
Defining Connections.....95
Compression Support for Cloud File Servers....146
Deleting a Connection.....147

5 - Spectrum Databases

Introduction to Spectrum Databases.....149
Installing a Spectrum Database.....149
Adding a Spectrum Database.....150
Database Pool Size and Runtime Instances.....150
Configuring database properties.....153
Deleting a Spectrum Database.....155

6 - Services

Spectrum Services.....157
External Web Services.....160

7 - Flows

Configuring Flow Defaults.....173
Scheduling Flows.....182
Viewing Flow Status and History.....185
Triggering a Flow with a Control File.....189
Command Line Execution.....193
Adding Flow Runtime Options.....210

8 - Performance

Performance Tuning Checklist.....214
Monitoring Performance.....228

9 - Monitoring

Email Notification.....235
Audit Log.....238

System Log....240
Notification Log....243
Logging the Record that Caused a Flow to Fail....243
Transaction Limit Warnings....244
Viewing Version Information....245
Viewing server status....245
Viewing and Exporting License Information....246
Logback configuration file....246

10 - Backup and Restore

About scheduled backups....249
Creating a Backup Manually....253
Restoring a Server....254
Restore a Spectrum repository backup....256

11 - Settings

Data Stewardship Settings....258
Context Graph Settings....268

12 - Administration Utility

Getting Started with the Administration Utility....272
Audit Log Information....276
Spectrum Business Glossary....278
Data Stewardship....280
Context Graph....281
Data Sources....304
Dataflows....318
Entities....326
Folders....327
Information Extraction....329
Jobs....339
Spectrum Lineage & Impact Analysis....349
Spectrum Machine Learning....350
Match Keys....352
Match Rules....355
Best of Breed Rules....358
Metadata Connections....361
Notification....363
Open Parser Cultures....367
Open Parser Domains....368
Performance Monitor....370
Permissions....373
Physical and Logical Models....374
Process Flows....385
Product Data....394
Profiles....399
Roles....404
Scorecard....409
Search Indexes....413
Services....422
Spectrum Databases....426
Service pool size....497
System....499
Tables....508
Tokens....511
User Accounts....513

13 - Spectrum properties

spectrum-container.properties reference....520
Configure proxy server settings....539

14 - About Spectrum Technology Platform

What Is Spectrum Technology Platform?....542
Enterprise Data Management Architecture....543
Spectrum Technology Platform Architecture....547
Modules and Components....551

1 - Getting Started

In this section

Configuring a New System....5
Accessing Spectrum Management Console....6
Starting and Stopping the Server....8
Installing the Client Tools....9
Network Ports....10
Configuring license and expiration notification....13

Configuring a New System

When you first install Spectrum Technology Platform there are few things you should do to ensure that your system has a basic level of security as well as access to the data you want to process through Spectrum Technology Platform.

1. Change the password for the admin user.

   Important: You should change the admin password immediately after installing Spectrum Technology Platform to prevent unauthorized administrative access to your system. Note that passwords, including those for the Web utilities, must contain one of the following special characters:

name	symbol
ampersand	&
asterisk	*
at	@
circumflex	^
dollar	$
double quote	"
hash/pound	#
percent	%
splat	!

   a) 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.

b) Log in using the default administrative credentials:

   User name: admin
   Password: admin

c) Go to **System** > **Security**.
d) Check the box next to the **admin** account then click the Edit button .
e) In the **New password** field, enter a new password. Enter it again in the **Confirm password** field.
f) Click **Save**.

2. Create users and roles as needed.

   For more information, see **Adding a User through Spectrum Management Console** on page 22.

3. Specify which folders on the Spectrum Technology Platform server you want to allow users to access.

   For more information, see **Limiting Server Directory Access** on page 47.

4. Decide if you want to allow Basic authentication for web service requests to your Spectrum Technology Platform server, or if you want to require token authentication. If you want to require token authentication, disable Basic authentication. For more information, see **Disabling Basic Authentication for Web Services** on page 56.
5. Define database resources if applicable.

   To determine if you need to define database resources, go to Spectrum Management Console **Resources** > **Spectrum Databases**. If you do not see the **Spectrum Databases** menu then you do not need to define database resources.

6. Define connections to the databases, file servers, and other data sources that you want to connect to from Spectrum Technology Platform. To define connections, go to Spectrum Management Console **Resources** > **Connections**.
7. Configure scheduled backups of your Spectrum Technology Platform server so that you can restore your server in the event of a severe system failure or other disaster. For more information, see **About scheduled backups** on page 249.

Accessing Spectrum Management Console

Spectrum Management Console is the tool for administering Spectrum Technology Platform. Using Management Console, you can perform such tasks as:

- Manage users and other security options
- Define connections to data sources such as databases or web services

- Configure remote component (database) properties
- Specify default settings for services
- Schedule job execution

To access Spectrum Management Console:

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. Enter a valid user name and password.

   The administrative user name is "admin" and it has a default password of "admin".

   Important: You should change the admin password immediately after installing Spectrum Technology Platform to prevent unauthorized administrative access to your system. Note that passwords, including those for the Web utilities, must contain one of the following special characters:

name	symbol
ampersand	&
asterisk	*
at	@
circumflex	^
dollar	$
double quote	"
hash/pound	#
percent	%
splat	!

Setting the Language and Region

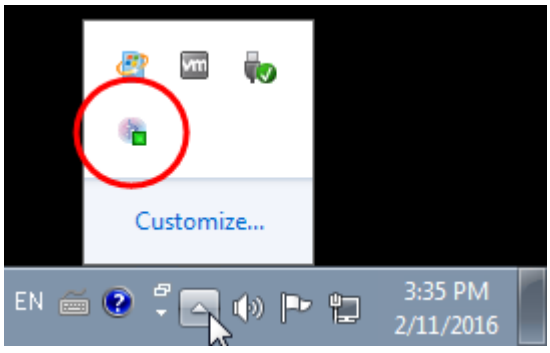
The Spectrum Management Console can be viewed in English, French, Japanese, German, and Spanish. It displays in the language setting of your browser as long as it is one of these languages. Otherwise, it displays in English. If your browser language or preference is unavailable, you can force one of the available languages to display by following these steps:

1. Log in to the Spectrum Management Console.
2. Click the user menu in the top right corner.
3. Select **Profile**.
4. In the **Language** field, choose the language you want.
5. In the **Country** field, choose your region. This setting controls the format to use when displaying dates, and times.
6. Click **Save**.

Starting and Stopping the Server

You may need to stop and start the Spectrum Technology Platform server to perform maintenance such as installing a lookup table or applying a product update.

- On Windows, Spectrum Technology Platform is set to start automatically when Windows starts up. To ensure that Spectrum Technology Platform has started, right click on the Spectrum Technology Platform icon in the Windows system task. If the icon is green then the server has started.



  To stop Spectrum Technology Platform, right-click the icon and select **Stop Spectrum**.
- To start the server on Linux:
  a) Change the working directory to the `bin` directory of where Spectrum Technology Platform is installed.

     For example:

```
cd /usr/g1/tst/server/bin
```

b) Source the setup file.

For example:

```
. ./setup
```

c) Start Spectrum Technology Platform.

- To start Spectrum Technology Platform in the background, type this command:

```
./server.start
```

- To start Spectrum Technology Platform in the foreground, type the following command:

```
./server.start console
```

- To stop Spectrum Technology Platform on Linux, type this command:

```
./server.stop
```

Note: Java uses `/var/tmp` as its temporary directory by default. If there is not enough space in this directory, the Spectrum Technology Platform server may not start.

Installing the Client Tools

The Spectrum Technology Platform client tools are applications that you use to administer your server and design and run dataflows and process flows. You must install your Spectrum Technology Platform server before installing the client tools.

Before installing, be sure to read the release notes. The release notes contains a list of known issues, important compatibility information, and release-specific installation notes.

This procedure describes how to install the client tools:

- **Enterprise Designer** allows you to create, modify, and run dataflows.
- **Job Executor** is a command line tool that allows you to run a job from a command line or script. The job must have been previously created and saved on Spectrum Technology Platform using Enterprise Designer or Flow Designer.
- **Process Flow Executor** is a command line tool that allows the execution of a process flow from a command line or script. The process flow must have been previously created and saved on Spectrum Technology Platform using Enterprise Designer or Flow Designer.
- **Administration Utility** provides command line access to several administrative functions. You can use it in a script, allowing you to automate certain administrative tasks. You can also use it interactively.

To install the client tools:

1. Open a web browser and go to the Spectrum Technology Platform Welcome Page at:

   `http://servername:port`

   For example, if you installed Spectrum Technology Platform on a computer named **myspectrumplatform** and it is using the default HTTP port 8080, you would go to:

   `http://myspectrumplatform:8080`

2. Click **Platform Client Tools** > **Desktop**.
3. If Microsoft .NET Framework 4.8 or later is not already installed on your Windows operating system, first click **Download Microsoft .NET Framework 4.8** to download and update Microsoft .NET Framework on the client machine.

   Versions of Microsoft .NET Framework 4.8 or later are included in Windows 10 updates starting from May 2019.
4. Download the client tool you want to install.

To install	Click
Enterprise Designer	**Desktop** > **Download Installer**
Job Executor	**Command Line** > **Job Executor** > **Download Jar**
Process Flow Executor	**Command Line** > **Process Flow Executor** > **Download Jar**
Administration Utility	**Command Line** > **Administration Utility** > **Download**
JDBC Driver	**Driver** > **JDBC Driver** > **Download Jar**

Related information

How to: Determine which .NET Framework versions are installed (Microsoft documentation)

Network Ports

The Spectrum Technology Platform server uses several network ports for communication. 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 Management Console. To troubleshoot the problem, look in the Spectrum Technology Platform log. This log shows which port is causing the problem. Open the following file to view the Spectrum Technology Platform server log:

SpectrumDirectory\server\logs\spectrum-server.log

Server port settings (spectrum-container.properties)

You can modify network ports by modifying properties settings in the following file:

SpectrumDirectory\server\conf\spectrum-container.properties

Restart the server after you make changes to the property settings.

Note: In a clustered environment you must modify the `spectrum-container.properties` file on *each node* in the cluster.

Port	Description
5001	This port is used by the Spectrum Technology Platform configuration database. To use a different port in a non-clustered environment, configure settings in the `repository/neo4j.template` file. To use a different port in a clustered environment, configure the **spectrum.repository.server.seeds** property to specify the seed nodes for the configuration database. See port **7687** for property changes for the configuration database.
5701	This port is used by Hazelcast for managing distributed processing between Spectrum Technology Platform servers in a cluster. To use a different port in a non-clustered environment, modify this property: `spectrum.cluster.port` To use a different port in a clustered environment: • Specify the port you want to use instead of 5701 in `spectrum.cluster.port`. • Include the Hazelcast port number after each IP address specified in `spectrum.cluster.seeds`. For example, if `spectrum.cluster.port` is set to 5702 and the IP address of a seed node is 1.2.3.4.5, you would specify `1.2.3.4.5:5702` in `spectrum.cluster.seeds`.
6001	TCP port number used to communicate between nodes in a Linux cluster. This must be open if required. Refer to configuration instructions for your cluster server environment.
6362	This port is used if you enable backups of the Spectrum Technology Platform configuration database. To use a different port, modify this property: `spectrum.repository.backup.port`

Port	Description
7474	This port is used by the Spectrum Technology Platform configuration database. To use a different port, configure settings in the *SpectrumFolder*\repository\neo4j.template file.
7687	This port is used by the Spectrum Technology Platform configuration database. To use a different port, configure the **spectrum.repository.port** property.
8080	The port used for communication between the server and Spectrum Enterprise Designer and Spectrum Management Console. This port is also used by web services. To use a different port, modify this property: spectrum.http.port
9200	This port is used by the index server. To use a different port, modify this property: spectrum.index.http.port
9300	This port is used by the search index engine used by the Advanced Matching, configured using spectrum.index.tcp.port.
10119	This port is used for API calls made to services. To use a different port, modify this property: spectrum.socketgateway.port
32751	This port is used for ODBC connections model stores which are created in Discovery. To use a different port, modify this property: spectrum.metadata.odbc.port

Context Graph port settings

Context Graph requires a separate installation of Neo4j Graph Database Server. For information about Neo4j Graph Database Server settings, see .

Context Graph connection settings for Neo4j Graph Database Server are configured in the Management Console in the Context Graph Repository Configuration settings. Port settings on the Neo4j Graph Database Server are specified in its neo4j.conf file.

The following default port settings are configured for use by Context Graph.

Port	Description
7700	This is the default port number that Context Graph uses for the Bolt Connector on the Neo4j Graph Database Server. This corresponds to the Bolt connector `dbms.connector.bolt.listen_address` and `dbms.connector.bolt.advertised_address` port settings in the `neo4j.conf` file of an instance of Neo4j Graph Database Server or of a node in a clustered configuration.

Machine Learning

Ports documented in this section are required for Machine Learning.

Port	Description
15431	Port 15431 is required for Machine Learning.

Configuring license and expiration notification

This procedure describes how to specify when to send expiration notifications, and the recipients of the notification emails.

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. Click **Configure Notification**.
4. Check the **Send notification** box.
5. In the **Days before expiration** field, specify the number of days in advance that you want to be notified of a pending license, software, or data expiration. This is the default value. You can specify a different notification period for each license item on the **System** > **Licensing and Expiration** page.

   For example, if you want to be notified 30 days before items expire, specify `30`.

6. Under **Recipients**, click the Add button + and enter the email address you want to receive the expiration notification email. You can multiple email addresses if needed.
7. Click **Save**.

You have now specified recipients for the notifications and how far in advance of expiration to send the notification email. If you have not already done so, you must configure a mail server to use to send the emails. Notifications will not be sent until a mail server has been configured.

> **Note:** By default the system will send expiration notifications for all items that expire (licenses, databases, software components, for example). You can disable expiration notifications for specific items by going to **System** > **Licensing and Expiration**.

2 - Approval Flows

In this section

Create an approval flow for an entity.....16
Create an approval flow for an Exception type.....17

Create an approval flow for an entity

The entity creation in Discovery follows a defined approval flow. The approval flow specifies:

- The levels of approval needed
- Roles for the various approval levels
- Days in which each approver in a specified level should revert with changes or approval

1. On **Spectrum Management Console** main menu, click **Resources** > **Approval Flows**.
   The **Approval Flows** page is displayed that shows categories of approval flows you can define.
2. Click to toggle the **Enabled** switch for **Discovery.Business Glossary Entity** to **ON**.
   This enables the flow you are going to define. This flow will be applicable to all glossaries.
   The **Approval Flow** window prompts whether to define approval levels.
3. Click **Yes** to define approval levels.
   This displays the **Edit Approval Flow** page. It shows that the **Entity type** is **Metadata Insights.Business Glossary Entity**.
4. To add an approval level, click the Add Approval Level button +.
   This adds a row to the table.
5. Enter these approval flow details:

Field	Description
Level	Defines the approval level. 1 denotes the first level of approval; 2, Second level of approval; and 3, Third level of approval.
Roles	Select the role that will act as the approver for this level. For example, if the level is 1, the selected role will act as the first level approver for all the glossary entities. **Note:** The roles displayed here are the ones the administrator has defined for glossary entities using the Spectrum Management Console **System** > **Security** > **Roles** option.
User	From the drop down list, select the default user for this approval level. **Note:** It is particularly useful in BSM, where records are allocated to a specific user rather than all mapped users of a role.
Days to Approve	Select the days within which an approver has to approve or suggest edits to the entity.

6. To add additional levels of approval, repeat steps 4 on page 16 through 5 on page 16.
7. Click **Save**.
   The approval flow is saved and the number of levels shows on the **Approval Flows** page. In the future, you can click its entry in the **Type** column to edit the approval flow.

Create an approval flow for an Exception type

An record in a workflow can be configured to follow an approval flow. The approval flow specifies:

- The levels of approval needed.
- Role for each approval level.
- Days in which each approver should accept or reject an Exception record.

1. On **Management Console** main menu, click **Resources** > **Approval Flows**.
   This displays the **Approval Flows** page. Exception types are labeled by **Exceptions.*typename***.
2. Click to toggle the **Enabled** switch for an Exception type to **ON**.
   This enables the flow you are going to define. This flow will be applicable when an exception condition has the selected Exception type.
   The **Approval Flow** window prompts whether to define approval levels.
3. Click **Yes** to define approval levels.
   This displays the **Edit Approval Flow** page. It shows that the **Entity type** is **Exceptions.*typename***.
4. To add an approval level, click the Add Approval Level button +.
   This adds a row to the table.
5. Enter these approval flow details:

Field	Description
Level	Shows the approval level. Level 1 denotes the data steward level of approval. Level 2 and above denote reviewer acceptance levels.

Field	Description
Roles	Select the role that will act as the approver for this level. **Note:** The roles displayed here are defined for an Exception type using Manage Console **System** > **Security** > **Roles** > **Data Stewardship** settings. All roles must have execute permissions to be included in an Exception approval flow. The role for the level 1 data steward reviewer should have modify in addition to view and execute permission. Roles defined for other levels must have view and execute permissions. A role may also have modify permission if you want to allow a user other than the data steward to edit Exception records.
Days to Approve	Select the days within which an approver has to edit, approve, or reject an Exception record.

6. To add additional levels of approval, repeat steps **4** on page 17 through **5** on page 17.
7. Click **Save**.
   The approval flow is saved and the number of levels shows on the **Approval Flows** page. In the future, you can click its entry in the **Type** column to edit the approval flow.

3 - Security

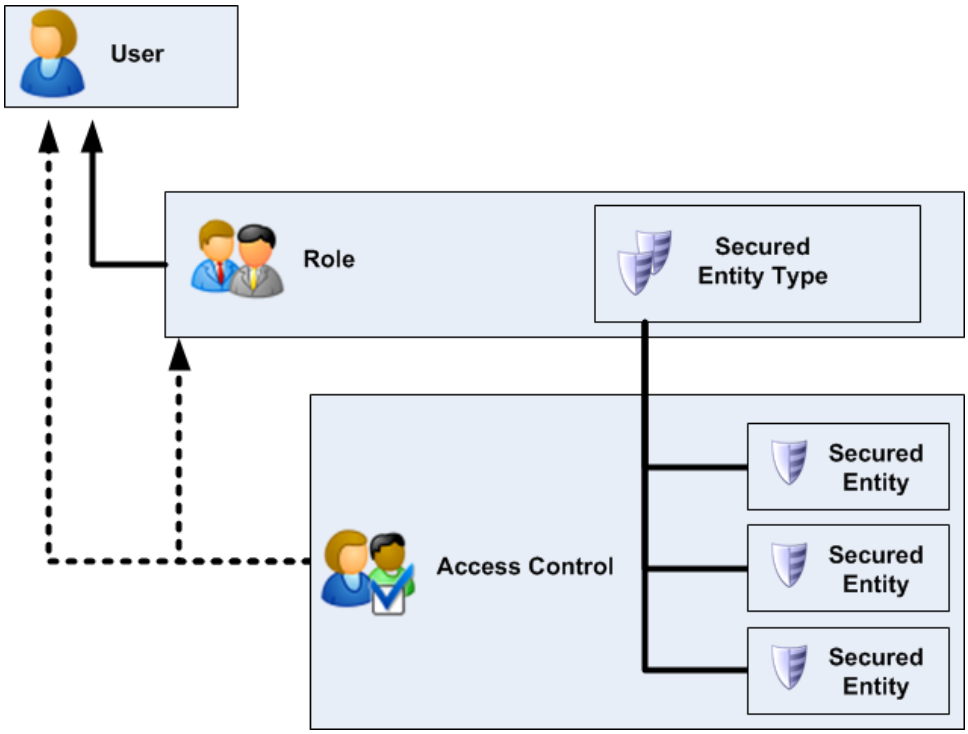
In this section

Security Model....20
Users....21
Roles....28
Access Control....38
Security for Spatial....41
Limiting Server Directory Access....47
Using HTTPS....49
Web Service Authentication....55
Using LDAP or Active Directory for Authentication....59
Implementing Spectrum Single Sign-on (SSO)....69
Encryption....77

Security Model

Spectrum Technology Platform uses a role-based security model to control access to the system.

The following diagram illustrates the key concepts in the Spectrum Technology Platform security model:



A *user* is an account assigned to an individual person which the person uses to authenticate to Spectrum Technology Platform, either to one of the client tools such as Spectrum Enterprise Designer or Spectrum Management Console, or when calling a service through web services or the API.

A user has one or more roles assigned to it. A *role* is a collection of permissions that grant or deny access to different parts of the system. Roles typically reflect the kinds of interactions that a particular type of user has with the system. For example, you may have one role for dataflow designers which grants access to create and modify dataflows, and another role for people who only need to process data through existing dataflows.

A role grants permissions to secured entity types. A *secured entity type* is a category of items to which you want to grant or deny access. For example, there is a secured entity type called "Dataflows" which controls the default permissions for all dataflows on the system.

If you need to fine-tune access you can optionally override the settings in the role or user by configuring access control. Access control settings work in conjunction with roles to define the permissions for a user. Roles define the permissions for categories of entities, such as all dataflows or all database

resources, and access control settings define the permissions for specific entities, called *secured entities*. Examples of secured entities include specific jobs or specific database connections. Defining access control settings is optional. If you do not define access control settings, the permissions defined in the role will control the user's permissions.

Access control settings work in conjunction with roles to define the permissions for a user. Roles define the permissions for categories of entities, such as all dataflows or all database resources, and access control settings define the permissions for specific entities, called *secured entities*. Examples of secured entities include specific jobs or specific database connections. For example, you may have a role that has granted the Modify permission to the secured entity type "Dataflows", but you may want to prevent users from modifying one specific dataflow. You could accomplish this by using access control to remove the Modify permission for the specific dataflow you do not want modified. You can specify access control settings for users and roles. Access control settings for a user override that specific user's permissions as granted by the user's roles. Access control settings for roles apply to all users who have that role.

Users

Spectrum Technology Platform user accounts control the types of actions users can perform on the system.

User accounts are required to:

- Use tools like Spectrum Management Console, Spectrum Enterprise Designer, Discovery, and command-line tools
- Run jobs on a schedule
- Run jobs from the command line
- Access services through web services or the API

There is an administrative account, **admin**, that comes with the system. This account has full access. The initial password is "admin".

Important: You should change the admin password immediately after installing Spectrum Technology Platform to prevent unauthorized administrative access to your system. Note that passwords, including those for the Web utilities, must contain one of the following special characters:

name	symbol
ampersand	&
asterisk	*

name	symbol
at	@
circumflex	^
dollar	$
double quote	"
hash/pound	#
percent	%
splat	!

You can create as many user accounts as you need.

Adding a User through Spectrum Management Console

This procedure describes how to create a Spectrum Technology Platform user account and assign a role to the account.

To add a user account and assign a role to it:

1. Open Spectrum Management Console.
2. Go to **System** > **Security**.
3. Click the Add button .
4. Leave the **Enabled** switch set to **On** if you want this user account to be available for use.
5. Enter the user name in the **User name** field.

   Note: User names can only contain ASCII characters. User names are case sensitive.

6. Enter the user's email address in the **Email address** field. The email address is used by some modules to send notifications to users.
7. Enter a description of the user in the **Description** field.
8. Enter and confirm the user's password.
9. Select the roles you want to give to this user.

   You may create your own roles or use one of the default roles. The default roles are:

   admin This role has full access to all parts of the system.

designer	This role is for users that create dataflows and process flows in Spectrum Enterprise Designer. It provides the ability to design and run dataflows.
integrator	This role is for users who need to process data through Spectrum Technology Platform but do not need to create or modify dataflows. It allows the user to access services through web services and the API, and to run jobs.
user	This is the default role. It provides no access to the system. Users who have this role will only gain access to the system if you grant permission through secured entity overrides.

For information about creating roles, see **Creating a Role in Spectrum Management Console** on page 28.

10. Click **Save**.

Changing a Password

This procedure describes how to change a user's password.

1. Open Spectrum Management Console.
2. Go to **System** > **Security**.
3. Select a user then click the Edit button .
4. Click **Change password**.
5. Enter the new password and enter it a second time to confirm it.
6. Click **Save**.

Related concepts

Users on page 21
Spectrum Technology Platform user accounts control the types of actions users can perform on the system.

Related tasks

Password Support Through JMX Console on page 26
JMX Console provides a set of attributes that give you control over the structure of your Spectrum Technology Platform system passwords.

Setting a Minimum Password Length

The minimum password length is enforced when creating or changing a password. Existing passwords that are shorter than the minimum length will continue to be valid.

To set a minimum password length:

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.config", click **com.pb.spectrum.platform.config:manager=AccountConfigurationManager**.
4. In the **updatePasswordPolicy** operation, set the **enableAdvanceControl** option to **True**.
5. In the **minLength** field, enter the minimum password length.
6. Click **Invoke**.
7. Click **Return to MBean View** to go back to the Account Configuration Manager screen.

Changing Your Email Address

The email address associated with your user account will receive notifications by some Spectrum Technology Platform modules.

To change your email address, follow these steps.

1. Log in to the Spectrum Management Console.
2. Click the user menu in the top right corner.
3. Select **Profile**.
4. In the **Email** field, enter your new email address.
5. Click **Save**.

Disabling a User Account

You have the option to disable a user account to prevent access to Spectrum Technology Platform.

Spectrum Technology Platform allows you to disable any user account except for the admin account. Jobs that run on a schedule using a disabled user account will not run.

Note: You cannot disable the user account "admin".

1. Open Spectrum Management Console.
2. Go to **System** > **Security**.
3. Check the box next to the user you want to modify then click the Edit button .
4. Switch the **Enabled** switch to **Off**.
5. Click **Save**.

The user account is now disabled and cannot access Spectrum Technology Platform.

Deleting a User through Spectrum Management Console

You have the option to permanently delete user accounts.

This procedure describes how to permanently delete a Spectrum Technology Platform user account.

Tip: User accounts can also be disabled, which prevents the system access without deleting the account.

1. Open Spectrum Management Console.
2. Go to **System** > **Security**.
3. Check the box next to the user you want to delete then click the Delete button .

   Note: The user account "admin" cannot be deleted.

Locking User Accounts

As a security precaution, user accounts are disabled after five unsuccessful authentication attempts in row. This includes unsuccessful authentication attempts to Spectrum Enterprise Designer, Spectrum Management Console, web services, and the Client API.

As an administrator, you can re-enable a user account by logging into Spectrum Management Console, editing the user, and switching the **Enabled** switch to **On**. User accounts can also be re-enabled using the Administration Utility. Users do not have the ability to unlock their own accounts.

Note: If you are using LDAP or Active Directory for authentication, the account locking rules of these services will apply. Your LDAP or Active Directory rules may allow more or fewer unsuccessful login attempts than Spectrum Technology Platform.

Unlocking the admin Account

Spectrum Technology Platform locks user accounts after several unsuccessful login attempts. You can unlock most locked user accounts through Spectrum Management Console, with the exception of the admin account. You must run a script on the server to unlock the admin account.

To unlock the admin account:

1. Log in to the server running Spectrum Technology Platform.

   If you are running Spectrum Technology Platform in a cluster, log in to any of the nodes. You only need to run the unlock script on one of the nodes.

2. Open a command prompt and go to the *`Spectrum Folder`*`\server\bin` folder.

3. (Linux only) Run this command:

   ```
   . ./setup
   ```

4. Run the enableadmin script:

   On Windows:

   ```
   enableadmin.bat -h HostAndPort -p AdminPassword [-s]
   ```

   On Linux:

   ```
   ./enableadmin.sh -h HostAndPort -p AdminPassword [-s]
   ```

   Where:

HostAndPort	The host name and HTTP port used by Spectrum Technology Platform. For example, `spectrumserver:8080`.
AdminPassword	The password for the admin account. If you do not know the admin account password and the admin account is locked, contact Precisely Technical Support.
-s	Specify `-s` if Spectrum Technology Platform is configured to use HTTPS.

Logging Out Inactive Sessions

Users of Spectrum Enterprise Designer and web clients such as Spectrum Management Console, Relationship Analysis Client, Data Stewardship, and others, are automatically logged out after 30 minutes of inactivity. You will see a warning message that your session will expire before you are logged out of the system.

Password Support Through JMX Console

JMX Console provides a set of attributes that give you control over the structure of your Spectrum Technology Platform system passwords.

Follow these steps to define your site's password requirements.

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. Under **Domain: com.pb.spectrum.platform.configuration**, select **com.pb.spectrum.platform.configuration:manager=AccountConfigurationManager**.

3. Set each attribute according to your site's requirements.

   Note: Click **set** to save each definition.

LowerCaseLetter — Specify true or false to require at least one lowercase letter. The default is false.

MinLength — Specify the minimum length for a password. The default is 6 characters.

Number — Specify true or false to require at least one number in a password. The default is false.

SpecialCharacters — Specify true or false to require at least one special character in a password. The default is false. Valid characters are:

name	symbol
ampersand	&
asterisk	*
at	@
dollar	$
percent	%
splat	!
question mark	?

UpperCaseLetter — Specify true or false to require at least one uppercase letter. The default is false.

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

You have defined global password requirements for Spectrum Technology Platform.

Roles

Spectrum Technology Platform comes with these some predefined roles.

A *role* is a collection of permissions that grant or deny access to different parts of the system. Roles typically reflect the kinds of interactions that a particular type of user has with the system. For example, you may have one role for dataflow designers which grants access to create and modify dataflows, and another role for people who only need to process data through existing dataflows.

Spectrum Technology Platform comes with these roles already defined:

admin	This role has full access to all parts of the system.
designer	This role is for users that create dataflows and process flows in Spectrum Enterprise Designer. It provides the ability to design and run dataflows.
integrator	This role is for users who need to process data through Spectrum Technology Platform but do not need to create or modify dataflows. It allows the user to access services through web services and the API, and to run jobs.
user	This is the default role. It provides no access to the system. Users who have this role will only gain access to the system if you grant permission through secured entity overrides.

To view the permissions granted to each of these roles, open Spectrum Management Console, go to **Security** and click **Roles**. Then select the role you want to view and click **View**.

Tip: You cannot modify the predefined roles. However, you can create new roles using the predefined roles as a starting point.

Creating a Role in Spectrum Management Console

A role is a collection of permissions that you assign to a user. If the predefined roles that come with Spectrum Technology Platform do not fit your organization's needs, you can create your own roles.

To create a role in Spectrum Management Console:

1. Open Spectrum Management Console.
2. Go to **System** > **Security**.
3. Click **Roles**.
4. Click the Add button .

   Tip: If you want to create a role that's similar to an existing role, you can make a copy of the existing role by checking the box next to the role you want to copy then clicking the Copy button . Then, edit the new role and continue with the steps below.

5. In the **Role name** field, enter the name you want to give to this role. The name can be anything you choose.
6. Click > corresponding to the Secured Entity group to which you want to grant permission. The group expands, showing all the secured entities.
7. Select the permissions you want to grant for each entity type. The permissions are:

   Create Allows the user to create entities that fall into this entity type's category. For example, if you allow the Create permission for the JDBC Connection entity type, users with this role would be able to create new database connections in Spectrum Management Console.

   View Allows the user to view entities contained by the entity type. For example, if you allow the View permission for the JDBC Connection entity type, users with this role would be able to view database connections in Spectrum Management Console.

   Modify Allows the user to modify entities contained by the entity type. For example, if you allow the Modify permission for the JDBC Connection entity type, users with this role would be able to modify database connections in Spectrum Management Console.

   Delete Allows the user to delete entities contained by the entity type. For example, if you allow the Delete permission for the JDBC Connection entity type, users with this role would be able to delete database connections in Spectrum Management Console.

   Execute Allows the user to initiate processing of jobs, services, and process flows. For example, if you allow the Execute permission for the Job entity type, users with this role would be able to run batch jobs. If you allow the Execute permission for the Service entity type, users with this role would be able to access services running on Spectrum Technology Platform through the API or web services.

8. Click **Save**.

The role is now available to be assigned to a user.

Warning: Granting Spectrum Spatial permissions to roles that you create on the Spectrum Management Console's **Roles** tab (on the **System** menu, select **Security**, and click the **Roles** tab) breaks permissions set in Spectrum Spatial Manager. Instead, create and manage permissions on specific resources or folders in Spectrum Spatial Manager. Setting Spatial permissions on a role created in the Spectrum Management Console gives that role full permission to read all named resources (including maps, layers, tables, and projects) and to edit all tables when granting data manipulation language (DML) permissions, which override permission set for the role in Spectrum Spatial Manager.

Deleting a Role in Spectrum Management Console

You can delete roles that are no longer used.

Delete roles that are no longer assigned to users.

Note: You cannot delete these roles: admin, user, designer, and integrator.

1. Open Spectrum Management Console.
2. Go to **System** > **Security**.
3. On the **Users** tab, make sure the role you want to delete is not assigned to any users. You cannot delete a role if it assigned to a user.
4. Click **Roles**.
5. Check the box next to the role you want to delete then click the Delete button .

Secured Entity Types - Advanced Matching

An entity type is a category of items to which you want to grant or deny access. The following entity types control access to parts of Spectrum Advanced Matching.

Best of Breed Rules Management	
Match Keys Management	
Match Rules Management	Controls access to the match rules in the Interflow Match stage, Intraflow Match stage, Transactional Match stage, and Match Rules Management Tool in Spectrum Enterprise Designer.
Search Index Management	Controls access to search indexes in Write to Search Index, Candidate Finder, and the Search Index Management Tool in Spectrum Enterprise Designer.

Secured Entity Types - Context Graph

After you establish roles, you can determine the permissions granted to each role. The following entity types control permissions to parts of Spectrum Context Graph.

Administration Controls the ability to perform these actions for Context Graph settings in Spectrum Management Console.

- View: Allows users to view Context Graph settings.
- Modify: Allows users to modify Context Graph settings.

Model Data Controls the ability to perform actions for Context Graph stages, Context Graph Visualization, and Context Graph Browser:

- View: Allows users to view a model.
- Create: Allows users to create models, including entities, relationships, and properties.
- Modify: Allows users to modify a models entity and relationship properties.
- Delete: Allows users to delete models, entities, relationships and properties.

> **Note:** This permission includes clearing a model in the Write to Model stage.

Model Metadata Controls the ability to perform actions for Context Graph stages, Context Graph Visualization, and Context Graph Browser:

- View: Allows users to view model metadata.
- Create: Allows users to create model metadata.
- Modify: Allows users to modify model metadata entity and relationship properties.
- Delete: Allows users to delete model metadata.

Query Controls the ability to perform actions for Context Graph Browser, Context Graph Visualization.

- Create: Allows users to create queries for models.
- View: Allows users to view queries for models.
- Modify: Allows users to modify queries for models.
- Delete: Allows users to delete queries for models.

Query Custom Script Controls the ability to use custom scripts to perform queries in Context Graph Visualization.

- Execute: Allow users to execute custom scripts.

Theme Controls the ability to perform the following actions in Context Graph Visualization.

- Create: Allows users to create themes for models.
- View: Allows users to view themes for models.
- Modify: Allows users to modify themes for models.
- Delete: Allows users to delete themes for models.

Secured Entity Types - Data Normalization

An entity type is a category of items to which you want to grant or deny access.

These entity types control access to parts of Spectrum Data Normalization.

Advanced Transformer Tables Controls access to the Advanced Transformer tables in the Table Management tool in Spectrum Enterprise Designer.

Open Parser Cultures	Controls access to cultures in the Open Parser Domain Editor tool in Spectrum Enterprise Designer.
Open Parser Domains	Controls access to domains in the Open Parser Domain Editor tool in Spectrum Enterprise Designer.
Open Parser Tables	Controls access to the Open Parser tables in the Table Management tool in Spectrum Enterprise Designer.
Standardization Tables	Controls access to the Standardization tables in the Table Management tool in Spectrum Enterprise Designer.

Secured Entity Types - Data Stewardship

An entity type is a category of items to which you want to grant or deny access. Approval flow types used to define approval flows are included here. The following permissions control access to Data Stewardship entity types.

Exceptions and approval flow types

Entity types control access to modify, delete, manage, and review exceptions in Data Stewardship. Approval flow types use execute permissions to allow types to be used to define approval flows. Approval flow types are created on the Spectrum Management Console **Data Stewardship Settings** page and listed in the table by name.

View Provides access to the Data Quality page to review exception trend data. A user who does not have View permissions will not be able to see the Data Quality page. For the Exception type, this setting provides access to view Exception records in an Approval flow.

Modify Provides access to the Manage page and all users exceptions on the Dashboard and Editor pages. A user who does not have Modify permissions will not have access to the Manage page or other users exceptions. For the Exception type, this allows a user to edit Exception records.

> **Note:** Modify permissions also controls access to users in the Read Exception stage. A user who does not have modify permission will not be able to read other users' exceptions from Data Stewardship.

Delete Provides access to the Purge section on the Manage page that allows a user to delete exceptions.

> **Note:** A user must also have Modify permissions to access the Manage page.

Execute Check this box to use approval flow types to create approval flows. See **Approval Flows** on page 15.

In addition to creating secured entity types, you can also use Access Control to create security entity overrides that specify exception record restrictions for specific dataflows or specific stages. These

overrides supersede user-based and role-based security entity type settings to make permissions more restrictive.

For example, if user JohnSmith has Modify permissions based on the secured entity type, but there are specific exceptions for a dataflow that his manager does not want anyone to alter, his manager can define an access control setting that restricts John Smith from modifying exceptions for that particular dataflow. John Smith is still able to modify other dataflows, but not the one for which there is the more restrictive access control setting.

Secured Entity Types - Database Resources

An entity type is a category of items to which you want to grant or deny access.

Database resources are available depending on which modules you have installed, for example:

- **Routing**
- **Spatial Database Resources**

Secured Entity Types - Discovery

An entity type is a category of items to which you want to grant or deny access. The secured entity types for Discovery control access to the Discovery web application's modeling, profiling, lineage, and impact analysis features.

Approval Flow	Controls access to the approval flow for Spectrum Business Glossary Entity.
Business Glossary Entity	Controls create, read, update, and delete operations [CRUD] for Business Glossary entities.
Business Glossary Semantic Type	Controls create, read, update, and delete operations [CRUD] for Business Glossary Semantic Type.
Lineage & Impact Analysis	Controls access to the **Lineage & Impact Analysis** view in Spectrum Discovery. Since **Lineage & Impact Analysis** only provides the ability to view information, the only permission available is **View**. **Note:** Giving users the **View** permission for **Lineage & Impact Analysis** will allow them to view logical models, physical models, and data stores even if they do not have the **View** permission for these secured entity types.
Lineage Impact Analysis Notes	Controls access to Notes in the **Lineage & Impact Analysis** view in Discovery.
Logical Model	Controls access to logical models in the **Model** section of Discovery.
Model Store	Controls access to model stores in the **Model** section of Discovery.

Physical Model	Controls access to physical models in the **Model** section of Discovery.
Profile	Controls access to profiles in the **Profile & Monitor** section of Discovery.
Query	Controls access to queries in the **Query** section of Discovery.
Query Schedule	Controls access to schedules in the **Query** section of Discovery.
Schedule	Provides access to schedules in the **Profile & Monitor** section of Discover
Scorecard	Provides access to scorecards in the **Profile & Monitor** section of Discovery.

Secured Entity Types - Enterprise Data Integration

An entity type is a category of items to which you want to grant or deny access.

There is one entity type that controls access to parts of Spectrum Data Integration.

Cache	Controls access to caches used by the Write to Cache and Query Cache stages, and the Cache Management tool in Spectrum Management Console.
Calendar	Controls access to calendars used by the Generate Time Dimension, and the Calendars tool in Spectrum Management Console.

Secured Entity Types - External Web Services

An entity type is a category of items to which you want to grant or deny access.

There is one secured entity type in the External Web Services category.

Connection	This secured entity type controls access to external web services in Spectrum Management Console and Spectrum Enterprise Designer.

Secured Entity Types - Platform

An entity type is a category of items to which you want to grant or deny access.

Example

Assume an entity type called "Dataflows," which controls permissions for all dataflows on the system. Platform entity types apply to all Spectrum Technology Platform installations, as compared to module-specific entity types that apply only if you have installed particular modules. The platform-level entity types are:

Audit Log	Controls access to the **System** > **Logs** > **Audit Log** area in Spectrum Management Console.
Dataflows	Controls access to all dataflow types (jobs, services, and subflows). **Note:** If a user does not have the Edit permission, the user will only see the exposed version and the last saved version in the **Versions** pane in Spectrum Enterprise Designer.
Dataflows - Expose	Controls the ability to make dataflows available for execution. **Note:** In order to expose the latest saved version of the dataflow (the version that's always at the top of the **Versions** pane in Spectrum Enterprise Designer) the user must have the Edit permission for the **Dataflows** secured entity type in addition to the Edit permission for the **Dataflows - Expose** secured entity type. This is because the latest saved version must first be saved as a version before it can be exposed, which requires the Edit permission for the dataflow.
Dataflows – Unlock	Controls access to unlock dataflows to make change.
Flow Defaults - Data Type Conversion	Controls access to the **Flows** > **Defaults** > **Data Type Conversions** area in Spectrum Management Console. All users have View access to data type conversion options. You cannot remove **View** access.
Flow Defaults - Malformed Records	Controls access to the **Flows** > **Defaults** > **Malformed Records** area in Spectrum Management Console. All users have View access to malformed record options. You cannot remove **View** access.
Flow Defaults - Reports	Controls access to the **Flows** > **Defaults** > **Reports** area in Spectrum Management Console. All users have View access to report options. You cannot remove **View** access.
Flow Defaults - Sort Performance	Controls access to the **Flows** > **Defaults** > **Sort Performance** area in Spectrum Management Console. All users have View access to sort performance options. You cannot remove View access.
Flow History - Jobs	Controls access to job execution history in Spectrum Enterprise Designer and Spectrum Management Console.
Flow History - Process Flows	Controls access to process flow execution history in Spectrum Management Console and Spectrum Enterprise Designer.
Flow History - Transactions	Controls access to the **Flows** > **History** > **Transactions** are in Spectrum Management Console.
Flow Scheduling	Controls access to the **Flow** > **Schedules** area in Spectrum Management Console.
Jobs	Controls the ability to run jobs in Spectrum Enterprise Designer, Spectrum Management Console, job executor, and the Administration Utility.
Notification - License Expiration	Controls access to configure license expiration notification emails in Spectrum Management Console.

Notification - SMTP Settings	Controls access to the **System** > **Mail Server** area in Spectrum Management Console.
Process Flows	Controls access to process flows. **Note:** If a user does not have the Edit permission, the user will only see the exposed version and the last saved version in the **Versions** pane in Spectrum Enterprise Designer.
Process Flows - Expose	Controls the ability in Spectrum Enterprise Designer to make process flows available for execution. **Note:** In order to expose the latest saved version of the process flow (the version that's always at the top of the **Versions** pane in Spectrum Enterprise Designer) the user must have the Edit permission for the **Process Flows** secured entity type in addition to the Edit permission for the **Process Flows - Expose** secured entity type. This is because the latest saved version must first be saved as a version before it can be exposed, which requires the Edit permission for the dataflow.
Product Data	Controls access to SPD-based product data.
Security - Access Control	Controls access to control settings in the **System** > **Security** > **Access Control** area in Spectrum Management Console.
Security - Access Token	Controls the ability to view users' tokens and delete tokens. A token facilitates authentication between a client and the server. • Read permission allows you to see a list of the active tokens, each of which represent an active session. • Delete permission allows you to delete users' tokens, which ends their session.
Security - Directory Access	Controls the ability to enable or disable restrictions on server directory resources using the **System** > **Security** > **Directory Access** area in Spectrum Management Console.
Security - Options	Controls access to the ability to turn security on and off in the **System** > **Security** > **Roles** area in Spectrum Management Console.
Security - Roles	Controls access to role configuration in the **System** > **Security** > **Roles** area in Spectrum Management Console.
Security - Users	Controls access for managing user accounts in the **System** > **Security** > **Users** area in Spectrum Management Console.
Security - Directory paths	Controls the ability to configure server directory resources the **System** > **Security** > **Directory Access** area in Spectrum Management Console.
Services	Controls the ability to run services through the API and web services.
Stages	Controls whether exposed subflows are available as a stage in dataflows.

System - Licensing	Controls access to the license information displayed in the **System** > **Licensing and Expiration** area in Spectrum Management Console.
System - Version Information	Controls access to the **System** > **Version** area in Spectrum Management Console.
System - Log	Controls access to the system log in Spectrum Management Console.

Secured Entity Types - Resource Connection

An entity type is a category of items to which you want to grant or deny access.

The Resource Connection entity types are:

Amazon DynamoDB	Controls access to all Amazon DynamoDB connections created using the **Connections** page in Spectrum Management Console and Discovery.
Amazon SimpleDB	Controls access to all Amazon SimpleDB connections created using the **Connections** page in Spectrum Management Console and Discovery.
Apache Cassandra	Controls access to all Apache Cassandra connections created using the **Connections** page in Spectrum Management Console and Discovery.
Connections	Controls access to all the datasource connections created in the Spectrum Management Console and Discovery.
File Server Connection	Controls access to all the file server connections created using the **Connections** page in Spectrum Management Console and Discovery.
Flat File	Controls access to all Flat File connections, whether Fixed Width or Delimited, created using the **Connections** page in Spectrum Management Console and Discovery.
JDBC Driver	Controls access to all the jdbc driver connections created using the **Connections** page in Spectrum Management Console and Discovery.
Marketo	Controls access to all Marketo connections created using the **Connections** page in Spectrum Management Console and Discovery.
Microsoft Dynamics 365	Controls access to all MS Dynamics CRM Online connections created using the **Connections** page in Spectrum Management Console and Discovery.
NetSuite	Controls access to all NetSuite connections created using the **Connections** page in Spectrum Management Console and Discovery.
Oracle Eloqua	Controls access to all Oracle Eloqua connections created using the **Connections** page in Spectrum Management Console and Discovery.
SAP	Controls access to all SAP NetWeaver connections created using the **Connections** page in Spectrum Management Console and Discovery.
Salesforce	Controls access to all Salesforce connections created using the **Connections** page in Spectrum Management Console and Discovery.
Splunk	Controls access to all Splunk connections created using the **Connections** page in Spectrum Management Console and Discovery.

Secured Entity Types - Spectrum Spatial

An entity type is a category of items to which you want to grant or deny access.

Spectrum Spatial has the following module-specific entity types:

Named Resources	Controls permissions to all named resources in Spectrum Spatial. Users of Spectrum Spatial services must have at least read permissions for the resources they use as well as for any dependent resources. When a named resource is created (using any tool, including Spectrum Spatial Manager, the Administration Utility, the Named Resource Service, and WebDAV), a new LocationIntelligence.Named Resource secured entity is automatically created for the named resource.
Dataset.DML	Controls permissions to datasets used in Spectrum Spatial that are associated with named tables. When a named table is created or uploaded (using any tool, including Spectrum Spatial Manager, the Administration Utility, the Named Resource Service, and WebDAV), a new LocationIntelligence.Dataset secured entity is automatically created for the associated dataset of that named table. A user must have View permissions on a named table *and* Create/Modify/Delete permissions on the dataset in order to perform DML operations on writable (JDBC-based) tables. DML operations include insert, update, and delete operations performed using the Write Spatial Data stage or the Feature Service.

Access Control

Access control settings work in conjunction with roles to define the permissions for a user. Roles define the permissions for categories of entities, such as all dataflows or all database resources, and access control settings define the permissions for specific entities, called *secured entities*. Examples of secured entities include specific jobs or specific database connections. For example, you may have a role that has granted the Modify permission to the secured entity type "Dataflows", but you may want to prevent users from modifying one specific dataflow. You could accomplish this by using access control to remove the Modify permission for the specific dataflow you do not want modified. You can specify access control settings for users and roles. Access control settings for a user override that specific user's permissions as granted by the user's roles. Access control settings for roles apply to all users who have that role.

Configuring Access Control in Spectrum Management Console

Access control settings work in conjunction with roles to define the permissions for a user. Roles define the permissions for categories of entities, such as all dataflows or all database resources,

and access control settings define the permissions for specific entities, such as specific jobs or specific database connections.

To configure access controls you must have View and Modify permissions to these secured entity types:

- Security - Access Control
- Security - Roles
- Security - Users

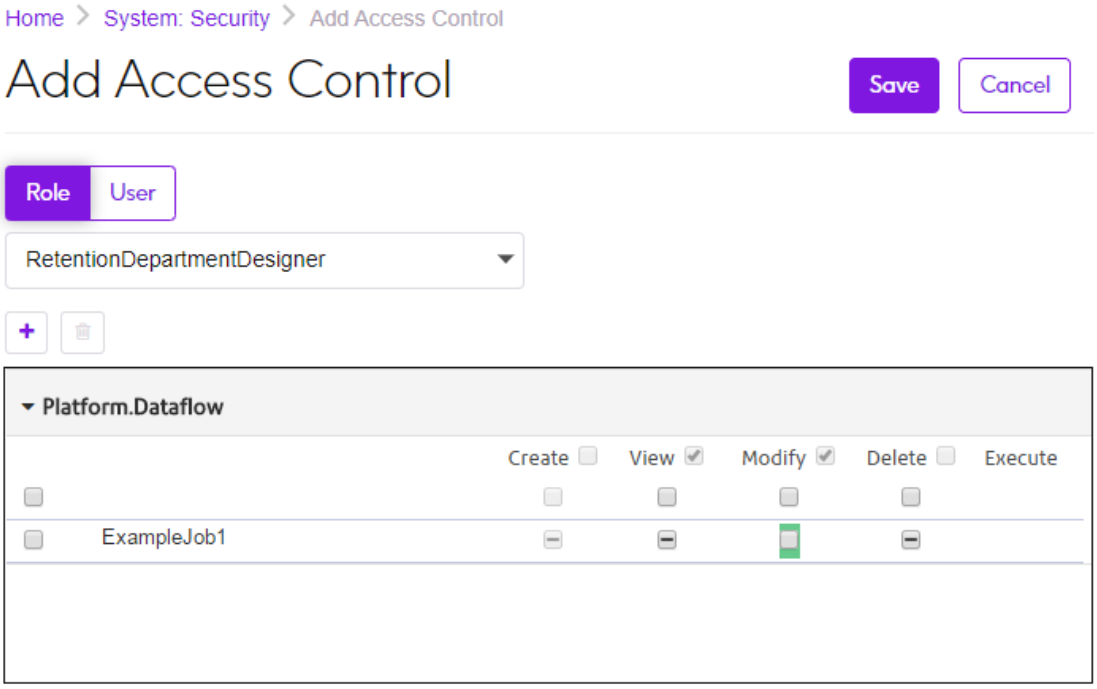
To configure access control:

1. In Spectrum Management Console, go to **System** > **Security**.
2. Click the **Access Control** tab.
3. Click the Add button .
4. Perform one of these actions:
   - If you want to specify access controls for a role, click **Role**. The access control permissions you specify will affect all users who have the role you choose.
   - If you want to specify access controls for a single user, click **User**. The access control permissions you specify will only affect the user you choose.
5. Select the role or user for which you want to define access controls.
6. Click the Add button .
7. Select the secured entity type that contains the secured entity you want. For example, if you want to configure access control for a dataflow, choose Platform.Dataflows.
8. Choose the secured entity you want to configure access controls for, then click the **>>** button to add it to the **Selected Entities** list.
9. Click **Add**.
   The secured entities you chose are displayed. The check boxes indicate the permissions in effect for the selected role or user.
10. Specify the permissions that you want to grant for each secured entity. Each secured entity can have one of these permissions:

	The permission is inherited from the role.
	The permission is inherited from the role and cannot be overridden.
	The permission is granted, overriding the permission specified in the user or role.
	The permission is denied, overriding the permission specified in the user or role.

Access Control Example

These are the access control settings for the role RetentionDepartmentDesigner.



In this example, the Platform.Dataflow secured entity type is set to allow the View and Modify permissions but not the Delete permission. So by default, any user that has the RetentionDepartmentDesigner role would have these permissions for all dataflows. However, you want to prevent users with this role from modifying the ExampleJob1 dataflow only. So, you clear the check box in the Modify column for ExampleJob1. Now users with this role will not be able to modify this dataflow but will still be able to modify other dataflows.

Multiple domain support

You have the option to define multiple domains in the **spectrum.ldap.dn.format** property of the ***SpectrumDirectory*/server/conf/spring/security/spectrum-config-ldap.properties** file.

To set multiple domains in spectrum.ldap.dn.format, use commas to separate each full domain name and provide the domain value in single quotes.

```
spectrum.ldap.dn.format='CN=%s,CN=Users,dc=spectest,dc=pvt',
'CN=%s,ou=Services,dc=spectest,dc=pvt','CN=%s,ou=managers,dc=spectest,dc=pvt'
```

Deleting Access Control Settings in Spectrum Management Console

When you delete access control settings for a user or role, the permission overrides defined by the access control settings are removed from the user or role.

For users, this means that the permissions granted by the user's role will take effect without any overrides. For roles, this means that the permissions defined in the role itself will take effect without overrides.

1. Open Spectrum Management Console.
2. Go to **System** > **Security**.
3. Click **Access Control**.
4. Check the box next to the user or role for whom you want to remove access control then click the Delete button .

Security for Spatial

Spectrum Spatial uses role-based security from the Spectrum Technology Platform. An administrator manages both users and roles in the Spectrum Management Console and assigns permissions, called Access Control List (ACL), to named resources in Spectrum Spatial Manager.

Users and Roles

Spatial administrators and sub-administrators grant permissions on roles and users to access or edit individual named resources and folders in the Spectrum Spatial repository.

The Spectrum Technology Platform Management Console has settings for managing users and roles. There are two kinds of roles that are relevant to Spectrum Spatial:

1. Predefined roles that are present when you install Spectrum. These confer certain default permissions to users who belong to them.
2. Custom roles that an administrator (admin) creates. A custom role has no permissions until specified in the Spectrum Spatial Manager.

Predefined Spatial Roles

After you install the Spectrum Spatial, four predefined roles are available in the Spectrum Management Console: two roles grant admin related privileges to users so they can manage content in Spectrum Spatial (spatial-admin and spatial-sub-admin), and two roles override resource permissions normally assigned in Spectrum Spatial Manager (spatial-user and spatial-dataset-editor).

spatial-admin

The spatial-admin role has full permissions to see and manage (view, create, delete, modify, and set permissions on) all content within the Spectrum Spatial repository. This role can edit data sets associated with named tables using the Feature Service (insert, update, and delete methods).

Users assigned to this role can log into Spectrum Spatial Manager, create new named connections, use the ACL REST API, use Map Uploader. The key difference between the spatial-admin role and the Spectrum Technology Platform admin role is that a spatial-admin cannot manage users or roles in Management Console.

spatial-sub-admin

The spatial-sub-admin role is similar to spatial-admin, but it cannot view all of the content within the Spectrum Spatial repository. This role views content in folders that it has read permission to. Users assigned to the spatial-sub-admin role must have permission to at least one folder.

Users assigned to this role can only manage (read, create, delete, modify, and set permissions on) folders that this role has to write permission to. However, a user may have more than one role, which means they can manage the folders those roles have permission on as well. The spatial-sub-admin role cannot edit datasets associated with named tables without granting additional permissions. They can log into Spectrum Spatial Manager, use the ACL REST API, and use Map Uploader, but only sees resources that are in the folders they have permission to.

The spatial-sub-admin role only has read access to named connections even when there is write access to folders in the named connection. This prevents these users from creating named connections in Spectrum Spatial Manager so they cannot browse the file system for files and create named tables circumventing the restrictions on what data they can see. To give the spatial-sub-admin role permission to create named connections, to delegate the connection creation, see **The Spatial Sub-Administrator Role and Named Connections**.

You can also assign users to the spatial-sub-admin role in Spectrum Spatial Manager.

spatial-user

The spatial-user role provides read permissions to all named resources in the Spectrum Spatial repository and overrides read permissions granted to named resources in Spectrum Spatial Manager. Do not assign users to this role if they require specific permissions.

Users assigned to this role can use the Spectrum Spatial web services to render tiles, maps, and layers and use the Feature Service to query tables. They cannot edit datasets associated with named tables. They do not have folder permissions, so they cannot manage resources.

spatial-dataset-editor

The spatial-dataset-editor role provides edit permissions (insert, update, and delete) to all datasets associated with named tables and overrides

permissions granted to named tables in Spectrum Spatial Manager. Do not assign users to this role if they require specific permissions.

Users assigned to this role can use the Spectrum Spatial Feature Service (insert, update, and delete methods) to edit and query tables. They do not have folder permissions, so they cannot manage resources.

Dataflow designers who are creating data flows must have a designer role (which is preset in Management Console). This is in addition to any permissions to access named resources, which are assigned by making them a member of spatial-user (so they can see all resources) or by using Spectrum Spatial Manager to grant permissions on specific named resources. For instructions on creating a spatial dataflow designer, see **Creating a Spatial Dataflow Designer** on page 44.

Custom Spatial Roles and Access Control Settings

Access control in Spectrum Spatial is managed using custom roles assign to users, which simplifies managing multiple users. Roles have specific permissions set. A user inherits the permissions of the roles that they are assigned. To specify permissions for access to specific named resources, use Spectrum Spatial Manager.

There are three kinds of permissions to view, edit, or manage data in Spectrum Spatial. We suggest creating roles for the following scenarios to grant:

- Read-only access to maps, layers, and tables available to the entire organization.

  Name this role **GeneralAcces**. All users may belong to this role, allowing any user in the organization to see these maps and layers.

- Read-only access to sensitive maps and layers.

  Add specific users to this role. Other users would not be able to see this data.

- Edit access to named tables.

  For example, you may have a table called Property Site Inspections that some users update, such as site inspectors who edit the data after visiting a property. You can grant edit permissions to this role and then assign your site inspectors to the role. Any other users viewing the table would not be able to edit the data.

- Write access to manage resources in a folder in the repository.

  As an example, you might create a role called SalesManagers with write permission to a folder in the Spectrum repository called `SalesData`. You could assign the spatial-sub-admin and SalesManagers roles to one or two users in the sales department. These users would then be able to use Spectrum Spatial Manager and the Map Uploader utility to manage named resources in the `SalesData` folder.

Creating a Spatial Dataflow Designer

The designer role provides access to Spectrum Technology Platform secured entity types, such as dataflows.

To create data flows, Spectrum Spatial stages, and services, a user must have a designer role and have execute permissions on the named resources in the data flows. The user should also be a spatial-user to see all resources, and the spatial-user role should have permission to see all resources in applications such as Spectrum Spatial Manager.

To assign the designer role to a user:

1. Open Management Console.
2. Go to **System** > **Security**.
3. Either select an existing user by clicking the Edit button, or click the Add button to create a new user.
4. In the **Roles section**, assign both the designer and spatial-user roles to the user account.
5. Click **Save**.

The user now has permission to view named resources and design dataflows using resources for Spectrum Spatial stages and services.

Limiting WebDAV Access to the Repository

WebDAV is a protocol to access and modify resources within the Spectrum Spatial repository but it can cause references between named resources and their Access Control List (ACL) to become inconsistent.

Use Spectrum Spatial Manager, the Named Resource service, and the Access Control service instead of WebDAV.

By default, accessing the repository using WebDAV is restricted to the machine where Spectrum Technology Platform is installed. You can provide access to servers by modifying the spatial java property file. You can do this by adding a property that includes a list (that is comma separated) of host names (IPs) that WebDAV is open to.

You must restart the Spectrum Technology Platform server to apply this change.

> **Note:** Limiting repository access using WebDAV applies to new Spectrum Technology Platform installations. The default value of the flag `repository.accesscontrol.allows` in `SpectrumDirectory/server/modules/spatial/spatial.properties` is blank.

To limit repository access using WebDAV:

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.configuration.properties**, click **com.pb.spectrum.platform.configuration.properties:manager=spatial.properties**.
4. Change the value of **repository.accesscontrol.allows**.

   Leaving the property empty disables all access to using WebDAV for all servers except the machine where Spectrum Technology Platform is installed.

   To allow other servers WebDAV access, enter a comma-separated list of server IP addresses. For example:

   ```
   192.168.2.1,192.168.2.2
   ```

5. Click the **Set** button next to the **repository.accesscontrol.allows**.
6. Restart the Spectrum Technology Platform server.

   For instructions on how to restart the Spectrum Technology Platform server, see the *Spectrum Administration Guide*.

Once this process is complete, WebDAV access is limited to the repository.

Using WebDAV with HTTPS

When communicating to the server over HTTPS to map a drive to the repository, a WebDAV client is required to use the TLS v1.2 protocol.

For client machines running on supported versions of Windows, you must apply a security patch and registry update to leverage this protocol.

1. On the client machine, apply the appropriate patch for the **operating system and version**.
2. Follow the instructions to update the registry to include support for TLS v1.2. The **DefaultSecureProtocols** value must be at least `0x00000800`.
3. Restart the client machine after changing the registry entry.

Encrypt a Jackrabbit Repository Password for Spectrum Spatial

These instructions will require restarting the Spectrum Technology Platform server.

The Password Management Utility generates the encrypted strings for a repository password by taking a password as input and encrypting it. This is a Spectrum Technology Platform utility that is located under `SpectrumDirectory/server/bin directory`. Supported entity types are `neo4j` for changing and encrypting a new password to the Spectrum Technology Platform Neo4j database and `spatial` for encrypting the password to the Spectrum Spatial Apache Jackrabbit repository.

For the Spectrum Spatial Jackrabbit repository there are two variations depending on whether the repository is held in:

- the standard Apache Derby database on the file system, which is the default setup after installation; or
- a fully configured relational database management system (RDBMS), such as SQL, Oracle, or PostGreSQL, as described in **Setting Up a Common Repository Database**.

The Password Management Utility only applies to the second case where a full RDBMS is used.

In the first case, the Apache Derby database is deployed in embedded mode, so that only the Spectrum Spatial Java virtual machine (JVM) can access it.

In the second case, you created a database user and password when setting up the RDBMS as the common repository database. We recommend using a strong password with over 10 characters that follows your organization's password rules. The Password Management Utility does not change the database password. It creates an encrypted version so that it is not stored in plain text on the Spectrum Technology Platform server.

To encrypt the password to Spectrum Spatial's Apache Jackrabbit database follow these steps:

1. Run the Password Management Utility to encrypt a repository password.
   a) Stop the Spectrum Technology Platform server.

      For instructions on how to stop the Spectrum Technology Platform server, see the *Spectrum Administration Guide*.

   b) On Windows, run a command prompt window as administrator (select **Run as administrator**).

      On Linux, open a Linux Terminal with execute permission to Spectrum Technology Platform file directories.

   c) Navigate to `SpectrumDirectory/server/bin`.
   d) On Windows, type `pmu.bat spatial`

      On Linux, type `pmu.sh spatial`

   e) At the prompt, enter the repository password to encrypt. The Password Management Utility uses the default encryption key for the Spectrum Technology Platform installation. There is no step for entering a custom key.

   To check the password encryption, open `SpectrumDirectory/server/modules/spatial/bootstrap.properties` in a text editor. The `bootstrap.properties` file includes the properties:

   - `repository.password.isencrypted` which indicates the jackrabbit repository password is encrypted and
   - `repository.password` which is the jackrabbit repository password.

Note: On a clustered environment, run the Password Management Utility on all nodes (that have the same password to encrypt) to update the `bootstrap.properties` file on each node. An encrypted copy of the password may vary between nodes, but the copy will decrypt to the original password.

2. Modify `SpectrumDirectory\server\modules\spatial\jackrabbit\repository.xml` to hold the password as a variable called `${rep.password}` as shown below for PostgreSQL. This is similar for SQL Server and Oracle.

```
<DataSources>
 <DataSource name="spatialrepo">
  <param name="databaseType" value="postgresql"/>
  <param name="driver" value="org.postgresql.Driver"/>
  <param name="url"
value="jdbc:postgresql://[server]:[port]/[database]"/>
  <param name="user" value="[user]"/>
  <param name="password" value="${rep.password}"/>
  <param name="validationQuery" value="select 1"/>
 </DataSource>
</DataSources>
```

Note: On a clustered environment, modify the `repository.xml` file on all nodes of the Spectrum Technology Platform configuration.

3. Restart the Spectrum Technology Platform server.

   For instructions on how to restart the Spectrum Technology Platform server, see the *Spectrum Administration Guide*.

The repository password is now encrypted to Spectrum Spatial's Apache Jackrabbit.

For information about Spectrum Technology Platform Neo4j database password encryption, see the *Spectrum Administration Guide*.

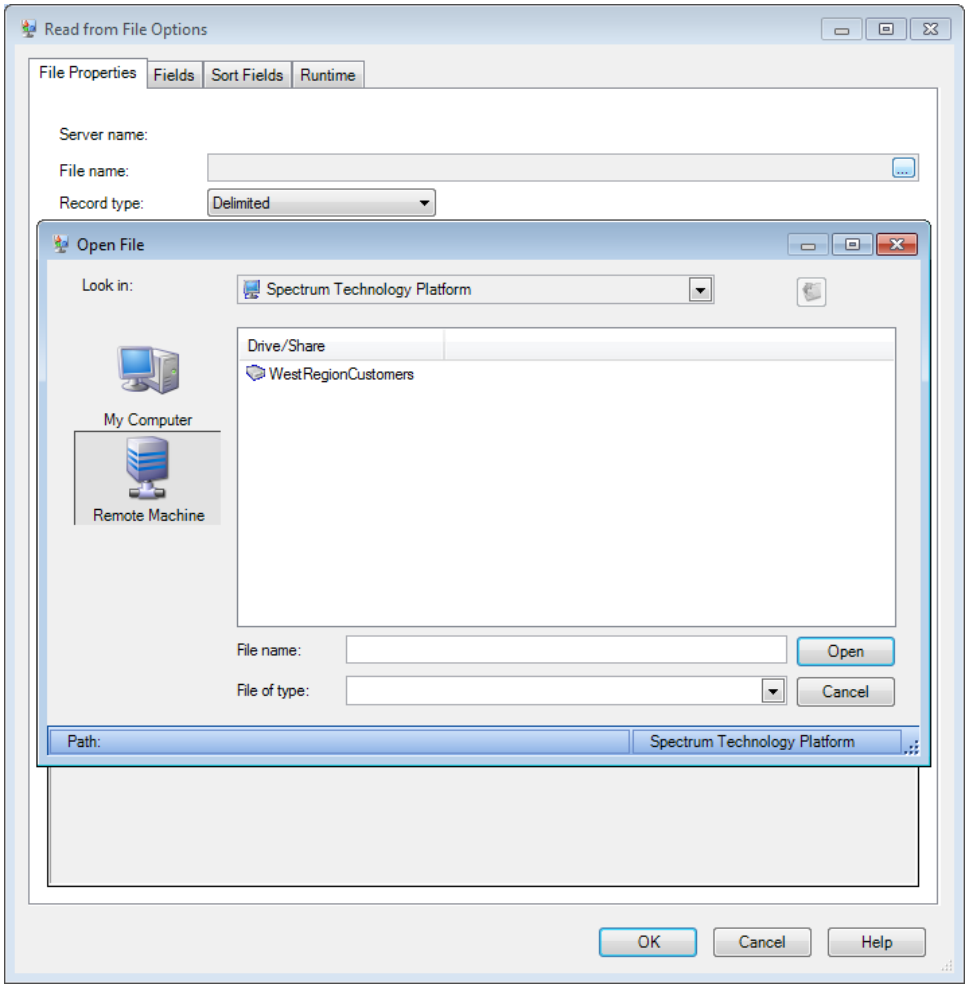
Limiting Server Directory Access

You can browse the Spectrum Technology Platform server's folders when performing tasks that require them to select a file. For example, users can browse the server when selecting an input or output file in a source or sink stage in Spectrum Enterprise Designer.

As an administrator, you may want to restrict access so that sensitive portions of the server cannot be browsed or modified.

One way to prevent access to the server's file system by making sure that users do not have the Platform security permission **Security - Directory Paths**. This prevents access to all folders on the server. You can also prevent access to some folders on the server while allowing access to others.

When you grant limited access, the folders you allow access to appear as the top-level folders in users' file browse windows. For example, if you allow users to only access a folder on the server named WestRegionCustomers, when users browse the server they would only see that folder, as shown here:



There are two situations where users can view the server's entire file system even if you have granted only limited access:

- When browsing for a database file while creating a Spectrum database in Spectrum Management Console
- When browsing for a JDBC driver file while creating a driver in Spectrum Management Console

To prevent users from browsing the server's entire file system, use roles to limit the user's access to Spectrum databases and JDBC drivers.

To provide access to some folders on the server while restricting access to others, follow this procedure.

1. Open Spectrum Management Console.
2. Go to **System** > **Security**.

3. Click **Directory Access**.
4. Set the **Limit access to server directories** switch to **On**.
5. Click the Add button .
6. In the **Name** field, give a meaningful name for the folder to which you are granting access.

   The name you provide here appears as the root name of the directory to users when browsing the server. In the example shown at the beginning of this topic, the name given to the accessible directory is WestRegionCustomers.

7. In the **Path** field, specify the folder to which you want to grant access. Users will be able to access all file and subfolders contained in the folder you specify.
8. Click **Save**.
9. If you want to grant access to additional folders, repeat the previous steps as needed.

Users now have access only to the folders you have specified. Note that users must have the Platform security permission **Security - Directory Paths** in order to access server directories.

> **Note:** If there are any dataflows that had previously accessed files that are no longer available because of file browsing restrictions, those dataflows will fail.

Using HTTPS

Secure Socket Layer (SSL) is a protocol that provides security for communications between client and server by implementing encrypted data and certificate-based authentication.

By default the Spectrum Technology Platform server uses HTTP for communication with Spectrum Enterprise Designer, browser applications such as Spectrum Management Console and Metadata Insights, as well as for handling web service requests and API calls. You can configure Spectrum Technology Platform to use HTTPS if you want to secure these network communications.

This section describes how to configure the Spectrum server to use Secure Socket Layer (SSL). Technically, the term "SSL" now refers to the Transport Layer ouSecurity (TLS) protocol, which is based on the original SSL specification. Before you configure HTTPS on the Spectrum server, you will need to create an RSA public/private key pair, generate a certificate signing request (CSR) that embeds your public key, share your CSR with a Certificate Authority (CA) to receive a final certificate or a certificate chain, and install the final certificate on the Spectrum server.

You can also also use a self-signed certificate, although this is only recommended for test purposes.

If you want to use HTTPS and you are running Spectrum Technology Platform in a cluster, do not follow this procedure. Instead, configure the load balancer to use HTTPS for communication with clients. Communication between the load balancer and the Spectrum Technology Platform nodes, and between the nodes themselves, will be unencrypted because Spectrum Technology Platform

clustering does not support HTTPS. The load balancer and the Spectrum Technology Platform servers in the cluster must be behind a firewall to provide a secure environment.

Enable HTTPS in Spectrum

By default the Spectrum Technology Platform server uses HTTP for communication with Spectrum Enterprise Designer, browser applications such as Spectrum Management Console and Metadata Insights, as well as for handling web service requests and API calls.

You can configure Spectrum Technology Platform to use HTTPS if you want to secure these network communications.

> **Note:** Spectrum Technology Platform uses TLS 1.2 to encrypt communication. Applications that access Spectrum Technology Platform web services or the API must support TLS 1.2 in order to connect over HTTPS.

This procedure describes how to enable HTTPS communication on a single-server installation of Spectrum Technology Platform. If you want to use HTTPS and you are running Spectrum Technology Platform in a cluster, do not follow this procedure. Instead, configure the load balancer to use HTTPS for communication with clients. Communication between the load balancer and the Spectrum Technology Platform nodes, and between the nodes themselves, will be unencrypted because Spectrum Technology Platform clustering does not support HTTPS. The load balancer and the Spectrum Technology Platform servers in the cluster must be behind a firewall to provide a secure environment.

To configure HTTPS communication for a single-server installation of Spectrum Technology Platform:

1. Stop the Spectrum Technology Platform server.
   - To stop the server on Windows, right-click the Spectrum Technology Platform icon in the Windows system tray and select **Stop Spectrum**. Alternatively, you can use the Windows Services control panel and stop the Precisely Spectrum Technology Platform service.
   - To stop the server on Linux, source the *SpectrumDirectory*`/server/bin/setup` script then execute the *SpectrumDirectory*`/server/bin/server.stop` script.
2. Load the certificate into a JSSE keystore. For more information, see **Loading Keys and Certificates** in the **Jetty 9 documentation**.
3. Using a text editor, open the file `spectrum-container.properties` located in *SpectrumDirectory*`/server/conf`.
   a) Uncomment and configure **Spectrum HTTP settings**:

```
##############################################
#Spectrum HTTP settings
##############################################
spectrum.http.default.protocol=https
spectrum.https.enabled=true
spectrum.https.port=8443
```

The `spectrum.keystore` and `spectrum.encryption` settings in this section should be configured to match your installation.

b) Configure settings in the **Spectrum SSL settings** section as required to match your installation.

c) Uncomment and configure hostname and port in the **Spectrum runtime settings**:

```
##############################################
#Spectrum runtime settings
##############################################
spectrum.runtime.hostname=fully qualified domain name
spectrum.runtime.port=8443
```

4. Import the certificates you are using. For example:

```
keytool -importkeystore -srckeystore
"C:\Precisely\Spectrum\server\conf\certs\keystore.p12" -destkeystore
"C:\Precisely\Spectrum\server\conf\certs\truststore.p12" -deststoretype
pkcs12
```

If you are using a self-signed certificate, see **Implementing self-signed certificates** on page 54.

5. If you are configuring HTTPS communication for Spectrum Spatial and services, you must perform an additional configuration prior to restarting the Spectrum Technology Platform server.

In Spectrum Spatial Manager, change the URLs in these service configurations to use HTTPS:

- 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 the *Managing Spatial* section of the *Spectrum Spatial Guide*.

6. Start the Spectrum Technology Platform server.

- To start the server on Windows, right-click the Spectrum Technology Platform icon in the Windows system tray and select **Start Spectrum**. Alternatively, you can use the Windows Services control panel to start the Precisely Spectrum Technology Platform service.
- To start the server on Linux, execute the *SpectrumDirectory*`/server/bin/server.start` script.

Generate Keys and Certificate Signing Requests

Before you configure settings on the Spectrum server, you must first generate keys and get a certificate signing request (CSR).

This section uses the OpenSSL command-line program to generate private/public keys and a CSR. OpenSSL comes with most Linux systems. For Windows, you can download an installation package for OpenSSL toolkit from **Shining Light Productions**. Refer to the **OpenSSL command reference** for more information about OpenSSL commands described here.

In the example code, replace references to *example* with your organization or domain name.

1. Open an OpenSSL Command Prompt.
2. Generate a public/private key pair using the an SSL toolkit..
   a) Execute the following command to generate an RSA public/private key pair.

      ```
      openssl genrsa -out www.example.com.key 2048
      ```

      This example generates an RSA 2,048 bit key pair. The encryption standard and minimum length requirements may differ depending on the version of Java installed on your machine and Certificate Authority requirements.

      This displays the following output:

      ```
      openssl genrsa -out www.example.com.key 2048
      Generating RSA private key, 2048 bit long modulus (2 primes)
      ..............................+++++
      ...+++++
      e is 65537 (0x010001)
      ```

   b) Generate a certificate signing request.

      ```
      openssl req -new -sha256 -key www.example.com.key -out
      www.example.com.csr
      ```

      This command embeds your public key and information about your organization into a certificate signing request or CSR. It will prompt for the following information:

      - Country Name (two-letter ISO-3166 country code)
      - State or province name
      - Locality name (such as city)
      - Organization name (such as company)
      - Organization unit name (unit or section in an organization)
      - Common name (typically the fully qualified domain name for the server)
      - Email address (contact address in your organzation)
      - A challenge password (be sure to make a record of this password)

   c) To ensure the validity of the CSR, run this command:

      ```
      openssl req -text -in www.example.com.csr -noout
      ```

The response should look something like this:

```
Certificate Request:
Data:
Version: 1 (0x0)
Subject: C = US, ST = Maryland, L = Rockville, O = Example Ltd, OU
 = Help Central Example, CN = www.example.com, emailAddress =
webmaster@example.com
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public-Key: (2048 bit)
Modulus:
   00:9f:b4:35:53:31:be:3b:79:42:23:0b:69:5b:10:
   ce:80:d0:a3:dd:4d:dc:b3:aa:c1:b8:38:b9:cb:d9:
   ...
Exponent: 65537 (0x10001)
Attributes:
challengePassword        :examplepassword
Signature Algorithm: sha256WithRSAEncryption
  02:a9:f0:2f:1b:7f:09:04:ff:6f:e4:5e:51:12:22:56:4c:5c:
  81:d4:e0:06:c2:0b:42:6d:18:de:29:df:97:d0:e2:24:53:35:
  ...
```

d) Submit the resulting CSR (`www.example.com.csr`) to a certificate authority.

Send the CSR to your CA, and follow their instructions to receive your final certificate. Different CAs charge differently for the service of vouching for your public key.

Using WebFolders to Access Spectrum Spatial Repository Resources

To add or modify a named resource, you can copy it to or from the repository using a WebDAV tool. Using WebFolders is an easy way to access the Spectrum Spatial repository and the resources contained in it.

Note: This task applies to Windows environments only.

To access the repository, you must be on the same machine where Spectrum Technology Platform and the repository are installed.

If you use WebDAV to make changes to named resources or metadata resource records such that they are not located in the same folder or do not have the same base name, then Spectrum Spatial Manager will no longer make matching changes to metadata records for move, rename or delete operations done on a resource.

To configure a WebFolder on Windows:

1. Using Windows Explorer, select `Map Network Drive...`

2. In the pop-up window, click on the link `Connect to a website...` to open the Add Network Location Wizard.
3. Click `Next` and select `Choose a custom network location`. Click `Next`.
4. In the `Internet or network address` field, add the repository URL. For example, `http://server:port/RepositoryService/repository/default/`. Click `Next`.
5. Enter your credentials (user name and password) if you are prompted for them.
6. Name the connection. For example, `Spectrum Spatial Repository`. Click `Next`.

Once finished, you will have a folder connection to the contents of the repository under your network places. You can use the WebFolder connection to the repository as you would any other Windows Explorer folder.

Implementing self-signed certificates

Spectrum SSL properties offer varying degrees of control of certificate verification through Certificate Authorities (CAs).

The role of a CA is to issue digital certificates to trusted entities and pass that trust to the SSL protocol that is trying to evaluate the certificate. If the CA cannot validate (trust) the entity, they can block authentication.

> **Note:** Although supported, we recommend against using self-signed certificates in a production environment. We do not consider this a best practice, as it overrides some authentication security checks.

SSL properties and defaults

Property/default	Description
`spectrum.encryption.selfSignedCert=false`	True or false: implement self-signed certificates in Spectrum Technology Platform
`spectrum.encryption.trustAllHosts=false`	True or false: implicitly trust all certificates; during verification, ignore host name specified on certificate
`spectrum.encryption.validateCerts=true`	True or false: bypass CA trust validation

Setting SSL handling and preferences for self-signed certificates

To implement self-signed certificates in Spectrum Technology Platform, first set this property in file **spectrum-container.properties**: `spectrum.encryption.selfSignedCert=true`.

Other SSL properties allow more specific, granular control of certificate verification through Certificate Authorities (CAs). The role of the CA is to issue digital certificates to trusted entities and pass that

trust to the SSLprotocol that is trying to evaluate the certificate. If the CA cannot validate (trust) the entity, they can block authentication.

• To bypass CA trust validation, you can set this property: `spectrum.encryption.validateCerts=true`.

• To implicitly trust certificates – signed or unsigned, and if the property `spectrum.encryption.validateCerts` is set to false, set this property: `spectrum.encryption.trustAllHosts`.

Related reference

spectrum-container.properties reference on page 520
This section provides a reference to properties in the `spectrum-container.properties` file, located in *`SpectrumDirectory`*`/server/conf/spectrum-container.properties`.

Web Service Authentication

Spectrum Technology Platform web services require authentication with valid user credentials. There are two methods for authenticating: Basic authentication and authentication by token.

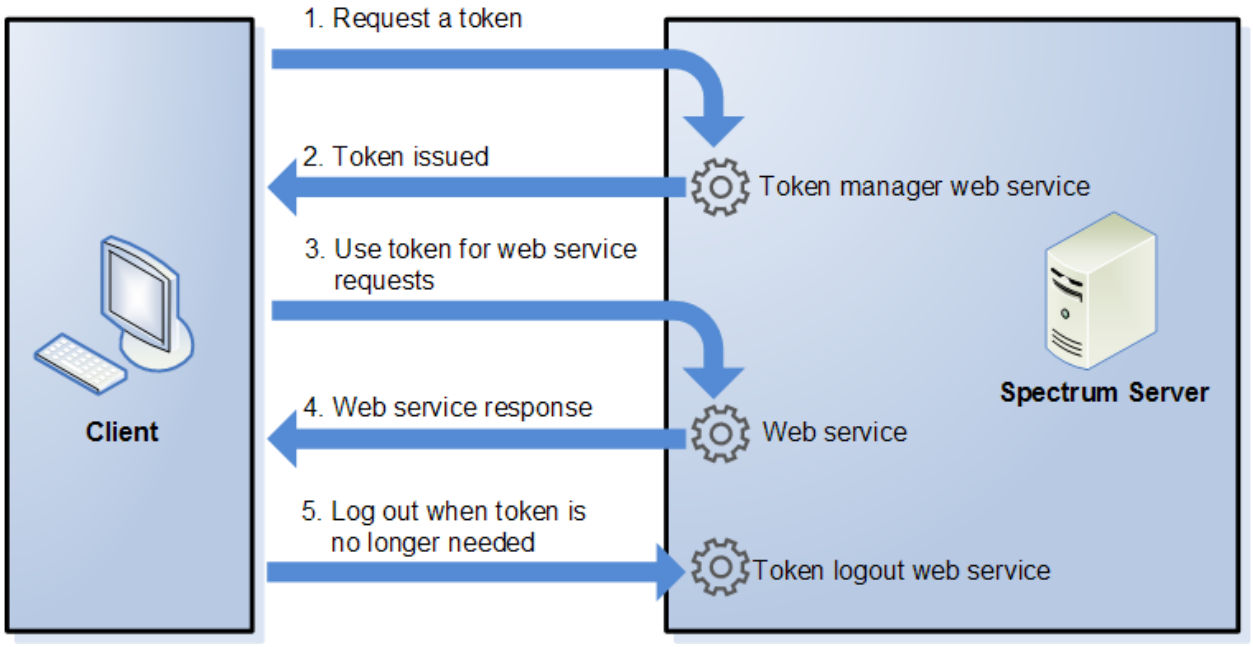
Basic authentication

With Basic authentication, the user ID and password are passed to Spectrum Technology Platform in the HTTP header of each request to the web service. Basic authentication is allowed by default, but your administrator may choose to disable Basic authentication. If Basic authentication is disabled you must use token authentication to access web services.

Authentication by token

With authentication with a token, the requester obtains the token from the Spectrum Technology Platform server, then uses it when sending a request to the web service. Instead of sending user credentials in each request, the token is sent to the server and the server determines if the token is valid.

The diagram below illustrates the process:



1. Obtain a token from the Spectrum Technology Platform server by sending a request to the token manager service.
2. The token manager service issues a token. If you requested a session token it also issues a session ID.
3. Send a request to the desired web service with the token in the HTTP header. For session tokens, include the session ID in the HTTP header.
4. The web service issues a response. You can use the token to make additional web service requests to either the same web service or any other web service on the Spectrum Technology Platform server. There is no limit to the number of web service requests you can make with a token, but if the token has an expiration limit (also known as a time-to-live) it will become invalid after the time-to-live has elapsed. If the token is a session token, it will become invalid after 30 minutes of inactivity.
5. When the token is no longer needed you should log out by sending a request to the token logout web service. This will remove the token from the list of valid tokens on the Spectrum Technology Platform server.

Disabling Basic Authentication for Web Services

Spectrum Technology Platform supports two types of authentication for web service requests: Basic authentication and token authentication. By default, both methods are enabled.

If you want to require web service requests to use token authentication instead of Basic authentication, you can disable Basic authentication by following these steps.

> **Note:** Be aware that disabling Basic authentication will cause existing clients to fail. For Spectrum Spatial, WMS, WMTS, and WFS clients will either be expecting Basic authentication

or no authentication. Leaving only token-based authentication will likely cause those clients to fail.

1. Stop the Spectrum Technology Platform server.
2. Open this file in a text editor:

   SpectrumDirectory/server/conf/spectrum-container.properties

3. Set this property to false:

```
spectrum.security.authentication.webservice.basicauth.enabled=false
```

4. Start the server.

Disabling Authentication for Web Services

Service-level authentication can be disabled for all SOAP or REST web services (or both). This is useful if you have your own high-level authentication built into the solution that is using, for example, Spectrum Spatial services.

All services and access to resources used by Spectrum Technology Platform are configured, by default, with authentication turned on.

To disable authentication for web services on the Spectrum Technology Platform :

1. Stop the Spectrum Technology Platform server.
2. Open the following file in a text editor:

   SpectrumDirectory\server\conf\spectrum-container.properties

3. Change the value of each property as needed. For example, to disable authentication for all SOAP services:

```
spectrum.security.authentication.webservice.enabled.REST=true
spectrum.security.authentication.webservice.enabled.SOAP=false
```

   Note: For Spectrum Spatial, REST services also include OGC web services.

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

Once finished, authentication is turned off for the type of web services that you specified.

Enabling CORS

Cross-Origin Resource Sharing (CORS) is a W3C standard that allows data sharing between domains. CORS enables web applications running in one domain to access data from another domain.

By enabling CORS on your Spectrum Technology Platform server, you can allow web applications hosted in another domain to access Spectrum Technology Platform web services.

For example, say you have a web application hosted at **webapp.example.com**. This web application contains a JavaScript function that calls a Spectrum Technology Platform web service hosted at **spectrum.example.com**. Without CORS, you would need to use a proxy server to facilitate this request, which would add complexity to your implementation. With CORS, you do not need to use a proxy server. Instead, you can designate **webapp.example.com** as an "allowed origin", thus permitting Spectrum Technology Platform to respond to web service requests that originate from the domain **webapp.example.com**.

To enable CORS on your Spectrum Technology Platform server:

1. Stop the Spectrum Technology Platform server.
2. Open this file in a text editor:

   SpectrumDirectory/server/conf/spectrum-container.properties

3. Edit the following parameters.

   spectrum.http.cors.enabled

   Set this property to true to enable CORS. The default is false.

   spectrum.http.cors.allowedOrigins

   A comma separated list of origins that are allowed to access resources on the Spectrum Technology Platform server. The default is http://localhost:8080,http://localhost:443, which allows access to resources using the default HTTP port 8080 and the default HTTPS port of 443.

   If an allowed origin contains one or more asterisks ("*"), for example http://*.domain.com, then asterisks are converted to .* and dots characters (".") are escaped to "\." and the resulting allowed origin is interpreted as a regular expression. Allowed origins can therefore be more complex expressions such as https?://*.domain.[a-z]{3} that matches http or https, multiple subdomains and any three-letter top-level domain (such as .com, .net, .org).

   spectrum.http.cors.allowedMethods

   A comma separated list of HTTP methods that are allowed to be used when accessing resources on the Spectrum Technology Platform server. The default value is POST,GET,OPTIONS,PUT,DELETE,HEAD.

   spectrum.http.cors.allowedHeaders

A comma separated list of HTTP headers that are allowed when accessing resources on the Spectrum Technology Platform server. The default value is X-PINGOTHER, Origin, X-Requested-With, Content-Type, Accept. If the value is a single asterisk ("*"), all headers will be accepted.

spectrum.http.cors.preflightMaxAge

The number of seconds that preflight requests can be cached by the client. The default value is 1800 seconds, or 30 minutes.

spectrum.http.cors.allowCredentials

Indicates whether the resource allows requests with credentials. The default value is true.

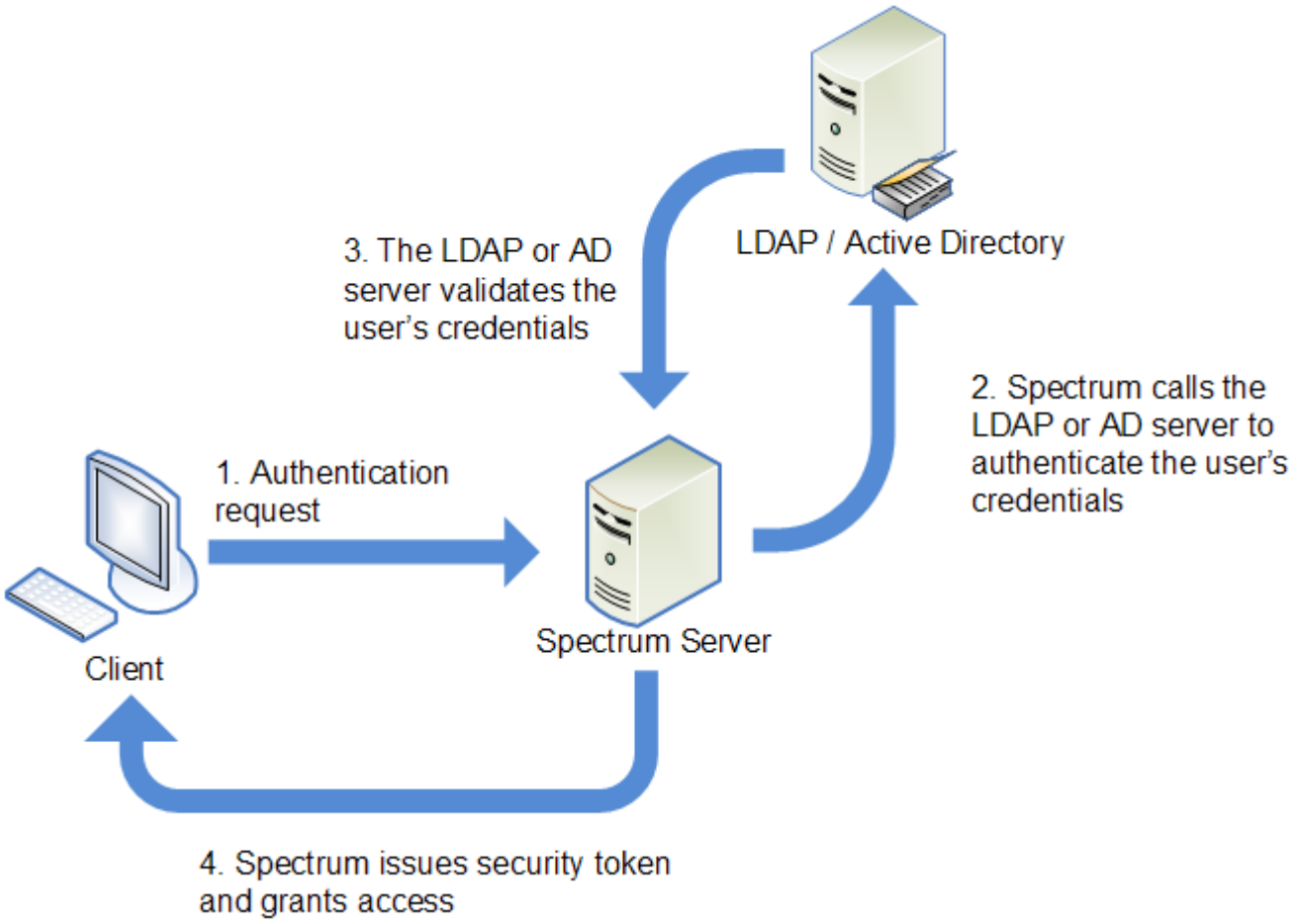
4. Save and close the file.
5. Start the Spectrum Technology Platform server.

Using LDAP or Active Directory for Authentication

Spectrum Technology Platform can be configured to use an LDAP or Active Directory server for authentication.

When a user logs in to Spectrum Technology Platform, the user's credentials are verified using LDAP or AD. The system then checks to see if there is a Spectrum Technology Platform user with the same name. If there is, the user is logged in. If there is not, then a Spectrum Technology Platform user account is automatically created for the user and given the role `user`.

Here is how the process works:



Before configuring Spectrum Technology Platform to use a directory service for authentication, confirm that your directory service meets these requirements:

- For LDAP, the directory server must be LDAP Version 3 compliant.
- There are no specific requirements for the Active Directory server.

Note: We recommend that you contact Precisely Technical Support or Professional Services to guide you through this process.

Note: When setting up Spectrum using LDAP or STS or SSO_STS , If the property is, by default, `spectrum.security.account.createNonExisting=true`, Active Directory users are created automatically in Spectrum Technology Platform after their first login to Spectrum. If you turn off the property `spectrum.security.account.createNonExisting=false`, LDAP/Active Directory users will not be authenticated to Spectrum Technology Platform until the administrator manually creates users.

1. If there are existing users configured in Spectrum Management Console and you want to use them after you enable LDAP or Active Directory authentication, create those users in your LDAP

or Active Directory system. Be sure to use the same user name as in Spectrum Technology Platform.

> **Note:** You do not need to create the "admin" user in LDAP or Active Directory since this user will continue to use Spectrum Technology Platform for authentication after you enable LDAP or Active Directory authentication.

2. Stop the Spectrum Technology Platform server.
3. Turn on LDAP or Active Directory authentication:
   a) Open this configuration file in a text editor:
   `server\conf\spectrum-container.properties`
   b) Set the property `spectrum.security.authentication.basic.authenticator` to LDAP:

```
spectrum.security.authentication.basic.authenticator=LDAP
```

   The setting `LDAP` is used to enable Active Directory as well as LDAP.

   c) Save and close the file.
4. Configure the connection properties:
   a) Open this configuration file in a text editor:
   `server\conf\spring\security\spectrum-config-ldap.properties`
   b) Modify these properties.

spectrum.ldap.url

The URL, including port, of the LDAP or Active Directory server. For example:
`spectrum.ldap.url=ldap://ldapserver.example.com:389/`

spectrum.ldap.dn.format

The format to use to search for user accounts in LDAP or Active Directory. Use the variable `%s` for the user name. For example,

LDAP:

```
spectrum.ldap.dn.format=uid=%s,ou=users,dc=example,dc=com
```

Active Directory:

```
spectrum.ldap.dn.format=%s@example.com
```

> **Note:** If you need to configure multiple LDAP domains, specify those domains separated by commas, with each domain value enclosed between

single quotes. This does not apply to single-domain configurations. For example:

```
spectrum.ldap.dn.format='CN=%s,CN=Users,dc=spectest,dc=pvt','
CN=%s,ou=Services,dc=spectest,dc=pvt','CN=%s,ou=managers,dc=spectest,dc=pvt'
```

spectrum.ldap.dn.base

The distinguished name (dn) to search for user accounts in LDAP or Active Directory. For example,

LDAP:

```
spectrum.ldap.dn.base=ou=users,dc=example,dc=com
```

Active Directory:

```
spectrum.ldap.dn.base=cn=Users,dc=example,dc=com
```

spectrum.ldap.search.filter

A search filter to use when searching for attributes such as roles. The search filter can contain these variables:

- `{user}` is the user name logging into Spectrum Technology Platform
- `{dn}` is the distinguished name specified in `spectrum.ldap.dn.base`.

For example, for LDAP:

```
spectrum.ldap.search.filter=uid={user}
```

And for Active Directory:

```
spectrum.ldap.search.filter=userPrincipalName={dn}
```

spectrum.ldap.attribute.roles

Optional. Specifies the LDAP or Active Directory attribute that contains the name of the Spectrum Technology Platform roles for the user. The role name you specify in the LDAP or Active Directory attribute must match the name of the role defined in Spectrum Technology Platform.

If this attribute contains a role named `designer` then the `designer` role would be granted to the user.

You can only specify one attribute but the attribute may contain multiple roles. To specify multiple roles inside an attribute, separate each with a comma. You can also specify a multivalue attribute, with each instance of the attribute containing a different role. Only the roles specified in this one attribute are used in Spectrum Technology Platform. No other LDAP or Active Directory attributes will have any impact on Spectrum Technology Platform roles.

If the user has roles assigned to it in Spectrum Technology Platform, the user's permissions are the union of the roles from LDAP or Active Directory and the roles from Spectrum Technology Platform.

For example, to apply the roles defined in the attribute `spectrumroles` you would specify:

```
spectrum.ldap.attribute.roles=spectrumroles
```

> **Note:** When a user logs in for the first time, if the user does not have a Spectrum Technology Platform user account one is created automatically and given the role `user`. The effective permissions for the user are the union of the permissions in the `user` role and the roles specified in the attributes listed in the `spectrum.ldap.attribute.roles` property.
>
> **Note:** When you view the user's roles in Spectrum Management Console you will not see the roles assigned to the user by the `spectrum.ldap.attribute.roles` property.

spectrum.ldap.pool.min

The minimum size of the connection pool for connections to the LDAP or Active Directory server.

spectrum.ldap.pool.max

The maximum number of simultaneous connections to the LDAP or Active Directory server.

spectrum.ldap.timeout.connect

Specifies how long to wait to establish a connection to the LDAP or Active Directory server, in milliseconds. The default is 1000 milliseconds.

spectrum.ldap.timeout.response

Specifies how long to wait for a response from the LDAP or Active Directory server after the connection is established, in milliseconds. The default is 5000 milliseconds.

spectrum.ldap.retry.count

The number of times the Spectrum Technology Platform server will try connecting to the LDAP or Active Directory server if the initial connection attempt fails. Set this to 0 if you want to allow only one connection attempt.

Tip: If you cluster your LDAP or Active Directory servers, we recommend that you set this value to 1 or more to allow the LDAP or Active Directory load balancer to redirect the connection request to a different server if the one that is initially tried is unavailable.

spectrum.ldap.retry.wait

The number of milliseconds to wait between connection attempts.

spectrum.ldap.retry.backoff

The multiplication factor to use to increase the wait time after each failed retry attempt. For example,

```
spectrum.ldap.timeout.connect=1000 ...
spectrum.ldap.retry.count=5
spectrum.ldap.retry.wait=500
spectrum.ldap.backoff=2
```

In this example, the wait for the initial connection attempt is 1,000 milliseconds, and the wait time for each of the five retry attempts is increased by a factor of two, resulting in these wait times for each retry attempt:

```
Retry attempt 1: 500 milliseconds
Retry attempt 2: 1,000 milliseconds
Retry attempt 3: 2,000 milliseconds
Retry attempt 4: 4,000 milliseconds
Retry attempt 5: 8,000 milliseconds
```

c) Save and close the properties file.

5. Start the Spectrum Technology Platform server.

If you are running Spectrum Technology Platform in a cluster, you must modify the `spectrum-container.properties` file and the `spectrum-config-ldap.properties` file on each of the servers in the cluster. Stop the server before modifying the file, then start the server after you are done modifying the file. If you mapped an LDAP attribute value to a role, this mapping will replicate to all nodes in the cluster, so you do not need to repeat the mapping procedure in the Spectrum JMX console.

Mapping LDAP Attribute Values to Roles

Set the property **spectrum.ldap.attribute.roles** to enable the mapping of attributes to user roles.

Before performing this procedure you must enable LDAP authentication. If you are using Spectrum Spatial, this also includes modifying the Jackrabbit configuration file. For more information, see **Using LDAP or Active Directory for Authentication** on page 59.

When you configure Spectrum Technology Platform to use LDAP or Active Directory for authentication, one of the configuration properties that you configure (the `spectrum.ldap.attribute.roles` property in the file `spectrum-config-ldap.properties`) specifies an LDAP attribute whose values determine the role to grant to a user. By default, the attribute values must match the Spectrum Technology Platform role names exactly in order for the role to be granted. For example to grant the designer role, the attribute you specify must contain the value designer.

If the LDAP attribute value that you want to use does not match the role name in Spectrum Technology Platform, you can map the LDAP attribute value to a role name. You can also map an LDAP attribute value that has the same name as a Spectrum Technology Platform role to a different role. For

example, one of the built-in roles is designer. If you have an LDAP attribute value named designer but you want it to map to another role, you could create a mapping.

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. Click this property:

   com.pb.spectrum.platform.common.security.role:mappings=RoleMappings

   > **Note:** This property is only visible after you enable LDAP authentication and the server is fully started. If you have not enabled LDAP authentication, see **Using LDAP or Active Directory for Authentication** on page 59.

3. In the **addMapping** section, in the **ldapValue** field, enter the LDAP attribute value that you want to map to a Spectrum Technology Platform role.
4. In the **roleName** field, enter the Spectrum Technology Platform role that you want to map to the LDAP attribute value.
5. Click **Invoke**.

Users who have the LDAP attribute will now be granted the role you specified when they long in to Spectrum Technology Platform.

To remove a mapping, enter the LDAP attribute you want to unmap in the **ldapValue** field in the **removeMapping** section.

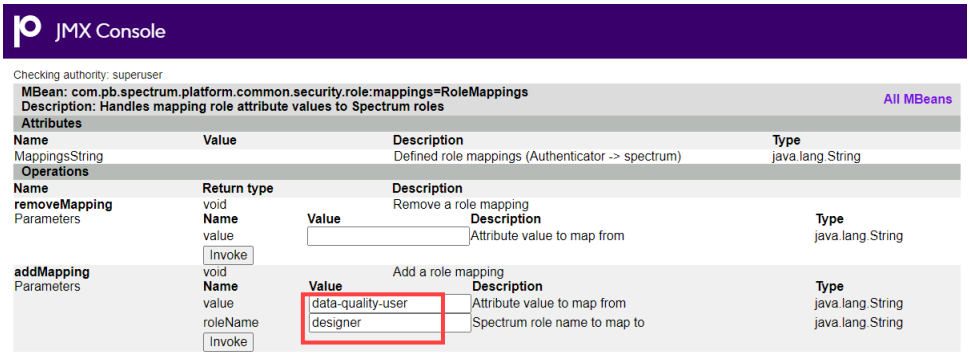
Example

Assume that you want to use a value in the `gecos` attribute to assign a role in Spectrum Technology Platform. If `gecos` contains the value data-quality-user, you want to grant the user the designer role when logging in to Spectrum Technology Platform.

To accomplish this, you would specify the `gecos` attribute as the attribute to use assign roles by specifying this in the file `spectrum-config-ldap.properties`:

```
spectrum.ldap.attribute.roles=gecos
```

Then, you would map the data-quality-user value to the designer role in the Spectrum JMX console:



As a result, any user that has the value data-quality-user in the `gecos` attribute will be granted the role designer.

Providing user login credentials in LDAP SSO installations

You can define internal authentication in LDAP SSO environments.

In LDAP SSO, the `spectrum.security.authentication.internal.users` property defines users that Spectrum will authenticate internally, as opposed to authenticating against an external LDAP AD FS user account repository. You must explicitly add user names to the `spectrum.security.authentication.internal.users` property in the spectrum-container.properties file. For example, if you do not define the "admin" user, that user cannot log in to the Spectrum Technology Platform server.

Add one or more users as follows:
`spectrum.security.authentication.internal.users=`*`user1`*`,`*`user2`*`,`*`user3`*

If someone tries to log in as "admin," and the admin user is not defined through this property, Spectrum will attempt to authenticate against LDAP, where that user may not exist. If you want to enforce external authentication through LDAP/AD FS, leave this property blank.

Enabling SSL Communication with LDAP

Communication between Spectrum Technology Platform and an LDAP or Active Directory server uses TCP by default.

You can configure Spectrum Technology Platform to use LDAP over SSL if you want to secure the communication between the Spectrum Technology Platform server and the LDAP or Active Directory server.

You may need to add the certificate to the Java TrustStore used by Spectrum Technology Platform if:

- The default Java TrustStore does not contain an entry for the certificate authority you are using.
- You are using a self-signed certificate. Note that using a self-signed certificate is not recommended in a production environment.

If either of these situations applies to you, add the certificate to the Java TrustStore by following these steps:

1. Obtain a copy of the certificate. You can get a copy of the certificate from your LDAP administrator or by using a tool like LDAP Admin to view and save the certificate.
2. Add the certificate to a new or existing TrustStore using the `keytool` utility included in the JDK. For example:

   ```
   keytool -import -file X509_certificate_ldap.cer -alias
   server.example.com -keystore ldapTrustStore
   ```

   See your vendor's Java documentation for more information.

   Note: The certificate must meet the requirements for encryption and length for the version of Java used by Spectrum Technology Platform. To find out the version of Java, open Spectrum Management Console and go to **System** > **Version**. For more information, see **java.com/en/jre-jdk-cryptoroadmap.html**.

3. Stop the Spectrum Technology Platform server.
   - To stop the server on Windows, right-click the Spectrum Technology Platform icon in the Windows system tray and select **Stop Spectrum**. Alternatively, you can use the Windows Services control panel and stop the Precisely Spectrum Technology Platform service.
   - To stop the server on Linux, source the *SpectrumDirectory*`/server/bin/setup` script then execute the *SpectrumDirectory*`/server/bin/server.stop` script.
4. Open this file in a text editor:

   SpectrumDirectory`\server\conf\spring\security\spectrum-config-ldap.properties`

5. Configure these properties:

   spectrum.ldap.url

   Specify the URL of the LDAP server. Be sure to specify the SSL port number, which is typically 636. For example:

   ```
   spectrum.ldap.url=ldap://server.example.com:636
   ```

   Note: Do not include a slash character at the end of the URL.

   spectrum.ldap.useSSL

   Specify true to enable SSL communication with LDAP.

   spectrum.ldap.trustStore

Specify the location of the TrustStore containing the certificate to use for SSL communication with LDAP. For example on Windows:,

```
spectrum.ldap.trustStore=file:D:\\Certs\\MyTrustStore
```

On Linux:

```
spectrum.ldap.trustStore=file://Certs//MyTrustStore
```

spectrum.ldap.trustStore.password

Specify the TrustStore password.

Important: If you are running Spectrum Technology Platform in a cluster, repeat this procedure on each server in the cluster.

Disabling SSL Communication with LDAP

If you have configured Spectrum Technology Platform to use SSL communication with LDAP or Active Directory and need to switch back to using TCP, follow this procedure.

1. Stop the Spectrum Technology Platform server.
   - To stop the server on Windows, right-click the Spectrum Technology Platform icon in the Windows system tray and select **Stop Spectrum**. Alternatively, you can use the Windows Services control panel and stop the Precisely Spectrum Technology Platform service.
   - To stop the server on Linux, source the *SpectrumDirectory*`/server/bin/setup` script then execute the *SpectrumDirectory*`/server/bin/server.stop` script.
2. Open this file in a text editor:

   SpectrumDirectory`\server\conf\spring\security\spectrum-config-ldap.properties`
3. Configure these properties:

   spectrum.ldap.url

   Change the URL of the LDAP server to use the TCP port rather than the SSL port. The default is 389. For example:

   ```
   spectrum.ldap.url=ldap://ldapserver.example.com:389/
   ```

   Note: You must include a slash character at the end of the URL.

   spectrum.ldap.useSSL

   Specify false to disable SSL communication with LDAP.

   spectrum.ldap.trustStore

Comment out this property.

spectrum.ldap.trustStore.password

Comment out this property.

Implementing Spectrum Single Sign-on (SSO)

Spectrum Technology Platform provides single sign-on (SSO), leveraging Active Directory Federation Services (AD FS), Ping Identity, and Azure Identity Provider (IDP).

AD FS/Ping Identity IdP enables SSO capabilities to multiple Web applications through a single Active Directory account. SSO allows your users to access any Spectrum Technology Platform Web-based services with one set of credentials. IdP allows the sharing of trusted party information, seamlessly, using cookie-based authentication.

The IdP administration tool (`adfs.msc`) is a Microsoft® Management Console (MMC) snap-in. It is used to add account and resource partners, map partner claims, add and configure account stores, and identify and configure federation-aware Web applications.

System Requirements

Current system requirements are available at our **support site**.

If you are new to Spectrum Technology Platform, it may be helpful to review these topics:

- **Enable HTTPS in Spectrum** on page 50
- **Network Ports** on page 10

Configuration assumptions and SSO deployment checks

We have designed Spectrum SSO to be seamless to end-users. To facilitate this seamless implementation, before you implement SSO, ensure that some specific security features are in place.

Work with your systems administrators to perform these checks:

The system administrator has deployed the federation server.	Microsoft® provides online references for **federation server deployment** and **verification**.
The system administrator has installed and configured the IdP server role.	Ensure that AD FS/Ping Identity is set up and configured for your processing environment. AD FS employs a configuration Wizard that helps with this process.

Your system includes a recognized load balancer.	For HTTP-level implementation of Spectrum SSO, you must terminate HTTPS at the load balancer level in Spectrum cluster considerations.
Change the server host names, as appropriate.	Each cluster in your configuration has a unique host name (computer name). Use best practices for naming your host machines — such as including the DNS in the name — so that they are easily identifiable and traceable.

Server configurations for authentication support

Spectrum Technology Platform requires you to define specific server-side properties and entities to support SSO authentication through IdP—Active Directory Federation Services (AD FS), Ping Identity, or Azure Identity Provider (IDP).

Prerequisites

Your Spectrum Technology Platform server must be HTTPS-enabled before setting up the configurations defined in this section. Review **Configure HTTP or HTTPS** on page 80 for more information.

Set security authentication

The security setup process requires you define the SSO authentication type and apply JCE policy files.

> **Note:** Ensure that all configurations are in place before setting these properties. See **Enable HTTPS in Spectrum** on page 50 for more information.

To set your authentication properties, locate the `spectrum-container.properties` file in *`SpectrumDirectory`*`/server/conf/` and set the following property to `LDAP/SSO`:

```
spectrum.security.authentication.basic.authenticator=LDAP/SSO
```

.

This property setting instructs SSO to redirect browser background Web applications. STS is used for the visible Spectrum applications, such as Web services and CLI.

You must also download and apply the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files of same version as the version of Java running on your Spectrum Technology Platform server. This helps with the encryption and decryption of messages between IdP and the Spectrum Technology Platform server. Extract the files, placing the JCE and the `*.jar` files (`US_export_policy.jar`, `local_policy.jar`) with the Java Runtime Environment files in the *`JavaLocation`*`/jre/lib/security` directory on the host machine for the Spectrum Technology Platform server.

Set server authentication properties

When you configure Spectrum Technology Platform to use AD FS_STS or AD FS_SSO for authentication, ensure that SSO is available through SSL/TLS. This is required for the authentication server.

Configure settings in the following file:

SpectrumDirectory\server\conf\spring\security\spectrum-config-sso-sts.properties.

This file contains configuration values for LDAP/SSO-specific properties. For SSO-enabled authentication, change the following property setting to `true` to enable SSO:

```
spectrum.security.authentication.web.sso.enabled=true
```

This property enables browser-based SSO for Web applications and STS for all other applications (client or Web services, for example). If this property is set to `false`, and the authenticator is LDAP/SSO, basic STS will be the default authentication protocol.

Set keystore configuration properties

Spectrum Technology Platform server needs to shake hands with the IdP server, requiring a private and public key pair.

Set up the trust keystore, key, and keystore password for your Spectrum Technology Platform host keystore. Configure these property settings:

- `spectrum.sso.sp.keystorePath=`*`KeyStorePath`*

  for example:

  `c:/saml2/AD FS/AD FS-java/tomcat-ssl.keystore`

Security properties

- `spectrum.sso.sp.keystorePassword=`*`KeyStorePassword`*
- `spectrum.sso.sp.privateKeyPassword=`*`PrivateKeyPassword`*
- `spectrum.sso.sp.alias=`*`KeyAliasUsedWhileCreatingCert`*

Manage AD FS session timeout properties

IdP has its own session management property. You also have the option isolate and control the Spectrum Technology Platform session timeout. As a best practice, we recommend defining both properties describe here with the same timeout values.

Use this property to configure the session timeout after a defined period of inactivity in seconds:

```
spectrum.sso.IdP.maximumAuthenticationLifetime=1800
```

Note that the Spectrum server session timeout property setting has higher precedence than the SSO property. The configuration for Spectrum server session timeout is in the `spectrum-container.properties` file. Configure the following timeout property where *`seconds`* indicates the number of seconds before the session ends:

```
spectrum.security.authentication.session.timeout=seconds
```

Setup SAML2 assertion

For SAML2 assertions, you must download your site's preferred SAML metadata for the IdP, and store it locally to generate requests.

This SAML metadata (XML) generates SAML log in and log out requests from Spectrum Technology Platform:

```
spectrum.sso.IdP.identityProviderMetadataPath=LocalPath/ADFSv2.0-FederationMetadata.xml
```

The service provider generates its own SAML2 data, which can verify that Spectrum Technology Platform is configured properly as a service provider:

```
spectrum.sso.sp.serviceProviderMetadataPath=localpath/SP-FederationMetadata.xml
```

The IdP requires a relying party, generally the service provider information. Spectrum Technology Platform must generate an identifier recognized by the IdP. This information is added in the SAML request and is sent to IdP from Spectrum Technology Platform. IdP is already configured with identifier: `spectrum.sso.sp.serviceProviderEntityId=`*`YourIdentifierForRelyingParty`*. This helps to verify trusted requests to IdP. For example:

```
https://US-5H19PH2-10.pbi.global.pvt/AD FS/trust
```

Set SSO binding properties

Bindings use artifact resolution protocol to resolve SAML identity provided messages by reference.

To set the bind/transport authentication for clients, locate the `spectrum-config-sso-sts.properties` file in *`SpectrumDirectory`*`\conf\spring\security` directory. Set the binding types using the following property

```
spectrum.sso.idp.destinationBindingType=Property for the binding type
configuration
```

For example:

```
spectrum.sso.idp.destinationBindingType=urn:oasis:names:tc:SAML:
2.0:bindings:HTTP-Redirect
spectrum.saml.sts.idp.type=ADFS
```

Table 1: Binding types

Binding type	Definition
REDIRECT	Allows SAML protocol messages to be transmitted within URL parameters: This binding is used when the SAML requester and responder need to communicate using an HTTP user agent as an intermediary, but there is no direct path between the two.
POST	Allows SAML protocol messages to be transmitted within the base64-encoded content of an HTML form control: Similarly to the REDIRECT binding, POST is used when the SAML requester and responder need to communicate using an HTTP user agent as an intermediary, but there is no direct path between the two. Additionally, this binding may be needed if the responder requires interaction with the user, such as authentication.
Artifact	Allows the SAML request, the SAML response, or both of these to be transmitted as a reference, using a small stand-in called an artifact: The HTTP Artifact binding is used when the SAML requester and responder need to communicate using an HTTP user agent as an intermediary, but the intermediary's cannot accept or allow an entire message to be sent through it.

For more information about SAML bindings, see **this resource**.

SSO in a clustered configuration

As part of the setup of SSO in a clustered environment, you must first have your clustering configuration in place.

To ensure a correct clustering configuration:

- Load balancer must be HTTPS-enabled to use SSO in a clustered setup using IdP.
- Define a domain entry in the host file of all nodes and load balancer. This maps the domain and IP address for each node to be recognized by Spectrum SSO. For example, in a three-node cluster configuration, you would define:

```
node1IP hostname
node2IP hostname
node3IP hostname
ADFSIP hostname
loadBalancerIP hostname
```

- Review the Spectrum Technology Platform documentation on setting up clusters for more information (**Clustered Architecture**).

Managing and mapping roles and properties

Spectrum SSO conveniently maps user accounts to admin-assigned credentials.

Spectrum Technology Platform grants users with the proper shares when they log in using their LDAP/SSO credentials. To remove role mapping, enter the LDAP attributes to unmap in the **value** field in the **removeMapping** section of the Spectrum JMX console.

Ensure that your users are defined to Spectrum Technology Platform with the appropriate credentials and permissions. If any user has a property setting of `spectrum.security.account.createNonExisting=false`, the user will not be recognized and will not be authenticated for SSO. User names must be created manually, by the system administrator. A user who does not exist in the external authentication repository will not be able to log in to Spectrum, even if the user is manually created in the Spectrum Management Console. Once the user is created in the external authentication repository, they can log in to Spectrum.

Set up the Admin role

Users may be mapped to admin roles. Mapped admin-level users will have the same privileges as Spectrum admin-level users, but they will display as non-admin users with basic user role privileges.

You can edit the user privileges on the Security page in Spectrum Management Console to display true privileges. Default admin share and user roles do not automatically apply under Spectrum SSO implementation. To apply and display user role permissions, you must set properties for any user that is mapped to the *domain* user group.

To establish system-wide access profiles, including that of Administrator ("Admin"):

1. Open the following file in a code editor:

   SpectrumDirectory`\server\conf\spring\security\spectrum-config-sso-sts.properties`

2. Set the dynamic property to apply admin group permissions at Spectrum server startup:

   ```
   spectrum.security.authentication.idpserver.admin.role=rolename
   ```

   where *`rolename`* is the group name for users who will inherit system-level admin permissions.

3. Log in to the Spectrum JMX console, and search for this property:

   ```
   com.pb.spectrum.platform.common.security.role:mappings=RoleMappings
   ```

   This property manages the mapping of roles to all user groups.

4. Define the following:

Parameter	Description
`addMapping`	In the `value` field, enter the SSO role value that you want to map to a Spectrum Technology Platform role.

Parameter	Description
`roleName`	Enter the Spectrum Technology Platform role that you want to map to the LDAP attribute value.

5. Click **Invoke**.
   Users who have the SSO role will now be granted the role you specified after they log in to Spectrum Technology Platform at least one time.
6. To remove a mapping, enter the LDAP attribute you want to unmap in the **value** field in the `removeMapping` section.

Assign the Admin role

The last step in mapping an admin role is to assign the admin role to certain users or user groups.

1. Edit the following file:
   SpectrumDirectory`/server/conf/spring/security/spectrum-config-sso-sts.properties`
2. Locate this property:
   `spectrum.security.authentication.IdPserver.admin.role=`*GroupName*
3. Provide the *GroupName* that requires the Spectrum Technology Platform admin role, such as "Domain Users."
4. Login as a user under the group name you assigned, then establish roles for other users.
   Go to **Security > Users > Roles**, or use the Role Mapping process described in **Mapping LDAP/SSO roles to Spectrum Technology Platform roles** on page 75.

Mapping LDAP/SSO roles to Spectrum Technology Platform roles

Before mapping roles, ensure that you have enabled LDAP/SSO authentication.

> **Note:** We have verified identity providers AD FS and Ping Identity for Spectrum Technology Platform.

When you configure Spectrum Technology Platform to use LDAP/SSO for authentication, by default, the role values must match the Spectrum Technology Platform role names, exactly in order, to grant the role. For example, to grant the designer role, the role you specify must be "designer."

> **Note:** If you are using Spectrum Spatial, you must also update the Jackrabbit configuration file. For more information see **Using LDAP or Active Directory for Authentication** on page 59.

You can map non-matching LDAP/SSO role values to an existing Spectrum Technology Platform role name. You can also map an LDAP/SSO role value with the same name as a Spectrum Technology Platform role to a different role. For example, one of the built-in roles is "designer." If you have an LDAP/SSO role value that is also named "designer," but you want it to map to another role, you could create a role map.

To map an LDAP/SSO role value to an existing Spectrum role:

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. Select this property:

   `com.pb.spectrum.platform.common.security.role:mappings=RoleMappings`

   This property is visible only when you enable LDAP or LDAP/SSO authentication, and the Spectrum Technology Platform server is fully started.
3. In the **addMapping** section, configure these settings:
   a) In the **value** field, enter the LDAP/SSO role value to map to a Spectrum Technology Platform role.
   b) In the **roleName** field, enter the Spectrum Technology Platform role to map to the LDAP attribute value.
4. Click **Invoke**.

Users who have been assigned an LDAP/SSO role will now be granted the role you specified for them the next time they log in to Spectrum Technology Platform.

To remove a mapping, enter the LDAP attribute you want to unmap in the **value** field in the **removeMapping** section in Spectrum JMX console.

Related tasks

Enable HTTPS in Spectrum on page 50

By default the Spectrum Technology Platform server uses HTTP for communication with Spectrum Enterprise Designer, browser applications such as Spectrum Management Console and Metadata Insights, as well as for handling web service requests and API calls.

Encryption

Certificate-based encryption

Spectrum Technology Platform certificate-based encryption requires you to set up certain security, trust, and communication tools.

> **Note:** We suggest that you review all of the Spectrum Technology Platform encryption documentation, including the various **Encryption methods** on page 78, before you configure encryption for the first time.

By default, encryption is disabled for all portions of the server, including HTTP/HTTPS, indexing, and caching. Set encryption in **spectrum-container.properties**, using property `spectrum.encryption.enabled=true` to enable encryption for all portions of the server using the global settings.

If you prefer to set up one portion of the Spectrum Technology Platform for encryption, such as HTTPS for browser and API communication, ensure that `spectrum.encryption.enabled=false`.

Related concepts

Defining the keystore on page 78
Define the truststore that will store your trusted certificates and the keystores to store the private key components of your trust certificates.

Defining the truststore on page 78

Define the truststore that will store your trusted certificates and the keystores to store the private key components of your trust certificates.

Define the trust certificate

Spectrum requires you to define a trust certificate for all levels of encryption, storing that certificate in *SpectrumDirectory*/server/conf/certs.

Setup keystores and truststore

Define the truststore that will store your trusted certificates and the keystores to store the private key components of your trust certificates.

To enable encryption, you must define the truststore that will manage your trusted certificates and the keystores to store the private key components of your trust certificates. By default, Spectrum provides self-signed encryption certificates.

- keystore.p12 – This pkcs12 keystore contains the communication certificate chain.
- truststore.p12 – This pkcs12 keystore contains the root certificate authority (CA) public certificate.

Note: These certificates are not recommended for production.

You must place the keystores in the *SpectrumDirectory*/server/conf/certs folder.

Encryption methods

This section describes encryption methods, as well as their respective settings and properties.

We suggest that you review all of the available encryption methods before you set up encryption at your site.

- **Method 1: Configure Spectrum to accept user-provided CA certificates** on page 78
- **Method 2: Configure Spectrum with self-signed certificates provided by Precisely** on page 79
- **Method 3: Configure Spectrum with your own, self-signed certificates** on page 79
- **Separate configurations** on page 80

Method 1: Configure Spectrum to accept user-provided CA certificates

This configuration method accepts user-provided certificates that are certificate authority (CA) registered.

This is the recommended method, as it provides the highest level of security. For this configuration, all nodes of the same type (node or client) should have certificates with matching DNs, as shown below.

1. **Setup keystores and truststore** on page 78, and copy those to the *SpectrumDirectory*/server/conf/certs folder.
2. Set encryption settings in the server installation location:

   - spectrum.encryption.enabled=true
   - spectrum.encryption.algorithm=JASYPT

   - spectrum.encryption.selfSignedCert=false
   - spectrum.encryption.trustAllHosts=false
   - spectrum.encryption.keystoreType=pkcs12 or jks
   - spectrum.encryption.keystore=node-keystore.p12
   - spectrum.encryption.keystorePassword=*password*
   - spectrum.encryption.keystoreAlias=*keystore alias if one applies*
   - spectrum.encryption.truststoreType=pkcs12 or jks
   - spectrum.encryption.truststore=truststore.p12
   - spectrum.encryption.truststorePassword=*truststore password*

Method 2: Configure Spectrum with self-signed certificates provided by Precisely

This topic provides the steps to implement self-signed certificates from Precisely.

Note: This configuration is **not** recommended for production environments.

1. Stop the Spectrum server.
2. Change these properties in spectrum-container.properties:

   - spectrum.encryption.enabled=true
   - spectrum.encryption.algorithm=JASYPT
   - spectrum.encryption.selfSignedCert=true
   - spectrum.encryption.trustAllHosts=true

3. Start the Spectrum server.

Method 3: Configure Spectrum with your own, self-signed certificates

This configuration is not recommended for production environments.

1. Stop the Spectrum server.
2. Create the keystore and truststore and copy to the *SpectrumDirectory*/server/conf/certs folder. This is the required location.
3. Set encryption settings in the server location, *SpectrumDirectory*/server/conf
4. Change these properties in **spectrum-container.properties**:

   - spectrum.encryption.enabled=true
   - spectrum.encryption.algorithm=JASYPT

- `spectrum.encryption.selfSignedCert=true`
- `spectrum.encryption.trustAllHosts=true`

  Note: Set `spectrum.encryption.trustAllHosts` to `true` only if a single certificate will be used across multiple hosts.

- `spectrum.encryption.keystoreType=pkcs12 or jks`
- `spectrum.encryption.keystore=node-keystore.p12`
- `spectrum.encryption.keystorePassword=`*`keystorepassword`*
- `spectrum.encryption.keystoreAlias=`*`keystore alias, if one applies`*
- `spectrum.encryption.truststoreType= pkcs12 or jks`
- `spectrum.encryption.truststore=truststore.p12`
- `spectrum.encryption.truststorePassword=`*`truststorepassword`*

5. Start the Spectrum server.

Separate configurations

The configurations described in this section allow you to separately configure encryption protocols, caching, and indexing for portions of Spectrum Technology Platform.

We strongly suggest that you review **Certificate-based encryption** on page 77 to build a strong foundation for successful encryption setup. Remember: configurations for specific portions of Spectrum Technology Platform override global property settings.

Configure HTTP or HTTPS

You have the option to configure HTTP or HTTPS, using the settings in the **Spectrum http settings** section of **spectrum-container.properties**.

These settings allow you to enable or disable both or one of these protocols. If both HTTP and HTTPS are enabled, the **spectrum.http.default.protocol** property helps Spectrum to apply the correct protocol to use for internal communications.

Note: HTTP is enabled by default. HTTPS is not enabled by default.

For HTTPS or HTTP configurations, define a keystore and a truststore.

Related concepts

Defining the keystore on page 78
Define the truststore that will store your trusted certificates and the keystores to store the private key components of your trust certificates.

Defining the truststore on page 78

Define the truststore that will store your trusted certificates and the keystores to store the private key components of your trust certificates.

Configure caching

You can specifically configure broker entity and clustering properties.

Use the properties below to configure caching and remote internode calls. You will also want to review the specific **Caching settings** on page 85.

Broker settings

These settings control broker properties and are located in the "Spectrum broker settings" section of **spectrum-container.properties**.

Property	Description
`spectrum.broker.name`	Name of the broker entity; for example: SpectrumCluster-Broker
`spectrum.broker.port`	Port used by the broker, default is 5710
`spectrum.broker.password`	Hazelcast broker password; For more information: • Review **Generate encryption strings** on page 88. • Review **Encrypt passwords or mask encryption strings** on page 89.
`spectrum.broker.addresses`	IP addresses for all members of the cluster. If commented out, then processing defaults to the `spectrum.cluster.seeds` value.
`spectrum.broker.seeds`	IP address used as the sharing source for encryption. Typically, this value matches the `spectrum.broker.addresses` value. **Note:** If you comment out this value, processing defaults to the `spectrum.broker.address` value. If `spectrum.broker.addresses` is commented out, processing defaults to the `spectrum.cluster.seeds` value.

Cluster settings

These settings define cache cluster settings and are located in the "Spectrum cluster settings" section of **spectrum-container.properties**.

Property	Description
`spectrum.cluster.enabled`	Enable/disable cluster caching: true for enabled or false (default) for disabled
`spectrum.cluster.name`	Name of the cache cluster entity; for example: SpectrumCluster
`spectrum.cluster.password`	Hazelcast cache cluster password; For more information: • Review **Generate encryption strings** on page 88. • Review **Encrypt passwords or mask encryption strings** on page 89.
`spectrum.cluster.seeds`	Comma-delimited list of IP addresses for members of the cluster
`spectrum.cluster.port`	Port used by the cache cluster; default is 5701
`spectrum.cluster.nodeId`	For each node in the cluster, define a unique integer node ID

Configure indexing - Elasticsearch

You can specifically configure Elasticsearch indexing properties.

To configure properties for Elasticsearch, use properties defined in **Configure indexing - Elasticsearch** on page 82.

CLI encryption setup - Windows client only

These instructions are a template that you can apply to encryption definitions.

Apply these template instructions to encryption definitions for pflowexecutor, the enableadmin utility, and the Administration utility. In those cases, the properties files are labeled **pflowexecutor.properties**, **enableadmin.properties**, and **cli.properties**, respectively.

The CLI properties file is in the same directory as the CLI component's executable files. For example, if your jobexecutor.jar is located under `C:\Users\myUser\cliClients\jobexecutor`, place the properties file in the `jobexecutor` folder.

jobexecutor

For jobexecutor, create a properties file called **jobexecutor.properties**. In this example, you'll need copies of the Spectrum self-signed certificates located on the server in the certs folder: `node-keystore.p12` and `node-keystore/truststore.p12`. Copy those two files to a local directory, such as `C:\myKeys`.

```
# sample properties when using  a Spectrum self-signed cert
spectrum.encryption.algorithm=JASYPT
spectrum.encryption.keystoreType=pkcs12
spectrum.encryption.keystore=C:\\myKeys\\node-keystore.p12
spectrum.encryption.keystorePassword=p*********s
spectrum.encryption.keystoreAlias=spectrum
spectrum.encryption.truststoreType=pkcs12
spectrum.encryption.truststore=C:\\myKeys\\truststore.p12
spectrum.encryption.truststorePassword=p*********s
spectrum.encryption.truststoreAlias=spectrum
spectrum.encryption.trustAllHosts=true
spectrum.encryption.trustSelfSigned=true
```

enableadmin

To use enableadmin with SSL enabled, you must create a properties file, similar to that used for jobexecutor: **enableadmin.properties**. Precisely provides this file in `server/bin` that points to the certificates in the `server/conf/certs` folder.

Those properties are:

```
# enable admin account properties
spectrum.encryption.algorithm=JASYPT
spectrum.encryption.keystoreType=pkcs12
spectrum.encryption.keystore=../conf/certs/client-keystore.p12
spectrum.encryption.keystorePassword=p*********s
spectrum.encryption.keystoreAlias=spectrum-client
spectrum.encryption.truststoreType=pkcs12
spectrum.encryption.truststore=../conf/certs/truststore.p12
spectrum.encryption.truststorePassword=p*********s
spectrum.encryption.trustAllHosts=true
spectrum.encryption.trustSelfSigned=true
```

Encryption properties

This reference lists and describes the global and specific server portion encryption properties located in **spectrum-container.properties**.

Global encryption settings

Global encryption settings apply to all levels: http, https, model store, cache, and index. You can use the level-specific properties to define preferences at those specific levels.

Property	Description
`spectrum.encryption.enabled`	Enable or disable basic HTTP: true for enabled or false (default) for disabled **Note:** Spectrum encryption will evaluate and apply the global encryption settings even if this property is set to false, and will not allow Elasticsearch indexing unless the **Indexing settings** on page 87 are specifically applied.
`spectrum.encryption.algorithm`	Encryption algorithm to use for the resource password: JASYPT (default) or AES
`spectrum.encryption.keystoreAlias`	Alias of certificate, if applicable, or use first key found; for example `spectrum`
`spectrum.encryption.keystoreType`	Keystore type: pkcs12 (default) or jks
`spectrum.encryption.keystore`	Keystore file name in location *`SpectrumDirectory`*`/conf/certs`
`spectrum.encryption.keystorePassword`	Keystore password; For more information: • Review **Generate encryption strings** on page 88 • Review **Encrypt passwords or mask encryption strings** on page 89
`spectrum.encryption.selfSignedCert`	Are certificates self-signed? True or false
`spectrum.encryption.truststoreType`	Truststore type: pkcs12 or jks
`spectrum.encryption.truststore`	Truststore file name in location *`SpectrumDirectory`*`/server/conf/certs`

Property	Description
`spectrum.encryption.truststorePassword`	Truststore password; For more information: • Review **Generate encryption strings** on page 88 • Review **Encrypt passwords or mask encryption strings** on page 89
`spectrum.encryption.validateCerts`	Should certificates be validated? True (default) or false
`spectrum.encryption.trustAllHosts`	During verification, ignore host name specified on the certificate.

Caching settings

These definitions control caching settings and are located in the **Cache settings (Hazelcast)** section of **spectrum-container.properties**.

Property	Description
`spectrum.cache.encryption.keystoreType`	Keystore type: pkcs12 or jks
`spectrum.cache.encryption.keystore`	Keystore file name in *`SpectrumDirectory`*`/server/conf/certs`
`spectrum.cache.encryption.keystorePassword`	Keystore password. For more information: • Review **Generate encryption strings** on page 88 for information about generating encryption strings. • Review **Encrypt passwords or mask encryption strings** on page 89 for more information about encrypting or masking strings.
`spectrum.cache.encryption.truststoreType`	Truststore type: pkcs12 or jks
`spectrum.cache.encryption.truststore`	Truststore file name in *`SpectrumDirectory`*`/server/conf/certs`

Property	Description
`spectrum.cache.encryption.truststorePassword`	Truststore password; For more information: • Review **Generate encryption strings** on page 88 for information about generating encryption strings. • Review **Encrypt passwords or mask encryption strings** on page 89 for information about encrypting or masking strings.

HTTPS and HTTP settings

These definitions control settings to HTTP and HTTPS properties and are located in the "Spectrum http settings" section of **spectrum-container.properties**.

Property	Description
`spectrum.http.enabled`	Enable/disable basic HTTP
`spectrum.http.port`	HTTP port
`spectrum.https.enabled`	Enable/disable basic HTTPS: true or false
`spectrum.https.port`	HTTPS port
`spectrum.https.encryption.validateCerts`	Should certificates be validated?
`spectrum.https.encryption.trustAllHosts`	Trust all certificates if no keystore or truststore are provided?
`spectrum.https.encryption.selfSignedCert`	Are certificates self-signed?
`spectrum.https.encryption.trustAllHosts`	Is host name verification disabled?
`spectrum.https.encryption.keystoreType`	Keystore type: pkcs12 or jks
`spectrum.https.encryption.keystore`	Keystore file name in *`SpectrumDirectory`*`/server/conf/certs`

Property	Description
`spectrum.https.encryption.keystorePassword`	Keystore password For more information: • Review **Generate encryption strings** on page 88. • Review **Encrypt passwords or mask encryption strings** on page 89.
`spectrum.https.encryption.keystoreAlias`	Alias of certificate, if applicable, or use first key found
`spectrum.https.encryption.truststoreType`	Truststore type: pkcs12 or jks
`spectrum.https.encryption.truststore`	Truststore file name in *`SpectrumDirectory`*`/server/conf/certs`
`spectrum.https.encryption.truststorePassword`	Truststore password; For more information: • Review **Generate encryption strings** on page 88. • Review **Encrypt passwords or mask encryption strings** on page 89.

Indexing settings

These definitions control indexing settings and are located in the "Index settings (Elasticsearch)" section of **spectrum-container.properties**.

Property	Description
`spectrum.index.encryption.enabled`	Enable/disable encryption on indexing : true or false
`spectrum.index.encryption.trustAllHosts`	Is hostname verification disabled?
`spectrum.index.encryption.keystoreType`	Keystore type: pkcs12 or jks
`spectrum.index.encryption.keystoreAlias`	Alias of certificate, if applicable, or use first key found
`spectrum.index.encryption.keystore`	Index keystore name in *`SpectrumDirectory`*`/server/conf/certs`

Property	Description
`spectrum.index.encryption.keystorePassword`	Index keystore password in *`SpectrumDirectory`*`/server/conf/certs`; For more information: • Review **Generate encryption strings** on page 88. • Review **Encrypt passwords or mask encryption strings** on page 89.
`spectrum.index.encryption.truststoreType`	Index truststore type: pkcs12 or jks
`spectrum.index.encryption.truststore`	Index keystore name in *`SpectrumDirectory`*`/server/conf/certs`
`spectrum.index.encryption.truststorePassword`	Index truststore password; For more information: • Review **Generate encryption strings** on page 88. • Review **Encrypt passwords or mask encryption strings** on page 89.

Password algorithm setting

This definition controls password-level decryption settings and are located in **spectrum-container.properties**.

Property	Description
`spectrum.password.decryption.algorithm`	Encryption algorithm to use for decrypting the passwords: JASYPT (default) or AES

Generate encryption strings

There are two methods for generating encryption strings.

> **Note:** If you have a non-default password for your CA Signed Certificate/KEYSTORE FOR HTTPS configuration, you will only be able to use the encrypted string for the non-default password in Spectrum-Container properties. The server may not start if you use plain text for the for the non-default password.

Method one

Use the `*.jar` file utility, *`SpectrumDirectory`*`/server/bin/password-utility.jar` to generate the encryption string for the Spectrum default password.

Note: If you run the encryption of the same password multiple times, this will generate different strings. This provides additional encryption strength and security.

Generate an encrypted string for the default password, p****s.

Specify the command:

```
java -jar password-utility.jar -p password
```

where:

- *password* is your site's password

Sample output:

```
############################################################
############ Encrypted String for the password ############
############################################################
            9yOYoZ9W2aAF2Baapa5wIxCMNQ/9TZFP
```

Method two

You can also generate encryption strings from the Spectrum JMX console through the `encryptString` property. This request takes the format `MBean operation: invoke method encryptString on MBean` *`servername`*`:manager=EncryptTextManager`.

Encrypt passwords or mask encryption strings

You have the option to encrypt passwords and mask encryption strings so that sensitive information is not exposed in log files or displays.

To encrypt passwords or to mask encryption strings, update the following property files, replacing *p*********s* with your encryption string.

`spectrum-container.properties`

Configure these password encryption settings in the following location.

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

```
spectrum.encryption.algorithm=JASYPT
spectrum.broker.password=p*********s
spectrum.cluster.password=p*********s
spectrum.encryption.keystorePassword=p*********s
spectrum.encryption.truststorePassword= p*********s
#spectrum.https.encryption.keystorePassword=p*********s
#spectrum.https.encryption.truststorePassword=p*********s
#spectrum.cache.encryption.keystorePassword=p*********s
#spectrum.cache.encryption.truststorePassword=p*********s
#spectrum.index.password=p*********s
#spectrum.index.encryption.keystorePassword=p*********s
#spectrum.index.encryption.truststorePassword=p*********s
```

enableadmin.properties

Configure these password encryption settings in the following location.

SpectrumDirectory\Program Files\Precisely\Spectrum\server\bin\enableadmin.properties

```
spectrum.encryption.keystorePassword=p*********s
spectrum.encryption.truststorePassword=p*********s
```

jobexecutor.properties

This file is located on the client where CLI utilities are downloaded.

```
spectrum.encryption.keystorePassword=p*********s
spectrum.encryption.truststorePassword=p*********s
```

pflowexecutor.properties

This file is located on the client where CLI utilities are downloaded.

```
spectrum.encryption.keystorePassword=p*********s
spectrum.encryption.truststorePassword=p*********s
```

cli.properties

This file is located on the client where CLI utilities are downloaded.

```
spectrum.encryption.keystorePassword=p*********s
spectrum.encryption.truststorePassword=p*********s
```

Spectrum Platform Neo4j Password Encryption

Spectrum Platform stores data in a repository managed by an embedded Neo4j Graph Database Server. Password encryption helps protect against possible attackers or eavesdroppers. The encrypted password to the Neo4j repository is stored in the `spectrum-container.properties` file. The default password that Spectrum Platform Server uses to access the Neo4j repository is encrypted using a default key that is stored internally by the server. To enhance security, an administrator can use the Spectrum Technology Platform Password Management Utility to change the password and specify a new key for the Spectrum Platform Neo4j repository or the Spectrum Spatial Apache Jackrabbit repository. For instructions about how to use this utility to change the Neo4j repository password, see **Change the Neo4j repository password with optional new encryption key** on page 91. For instructions to change the Spectrum Spatial Apache Jackrabbit repository, refer to the *Spectrum Spatial Administration Guide*.

Change the Neo4j repository password with optional new encryption key

Use the Spectrum Platform Password Management Utility to change the password using an optional new encryption key for the Neo4j repository. The utility accepts a new password and encryption key from the user and generates and stores the encrypted string for the new password.

The Spectrum Platform Password Utility is located in the *`SpectrumDirectory`*`/server/bin` directory. Supported entity types are `neo4j` for encrypting the password to the Spectrum Platform Neo4j repository and `spatial` for encrypting the password to the Spectrum Spatial Apache Jackrabbit repository. This topic describes how to change the password to the Spectrum Platform Neo4j repository. Refer to the Spectrum Spatial Guide to encrypt a Jackrabbit repository password for Spatial.

1. Stop the Spectrum server.

   For instructions on how to restart the Spectrum server, see **Starting and Stopping the Server** on page 8.

2. Open a PowerShell window as administrator (select **Run as administrator**).
3. Depending on the configuration, navigate to the following `bin` directory.

Configuration	Navigate to the following directory
Server and database on same machine	*`SpectrumDirectory`*`/server/bin`
Database-only machine	*`SpectrumDirectory`*`/repository/bin`
Server-only machine	*`SpectrumDirectory`*`/server/bin`

4. Type the following command :

Operating system	Command
Windows	`pmu.bat neo4j`
Linux	`pmu.sh neo4j`

5. When prompted, enter the following information:

Prompt	Enter
`Enter Old Password:`	Enter the existing password for the Neo4j server. This prompt does not appear if the old password is the initial default password.
`Enter New Password:`	Enter the new password for the Neo4j server. New password should be at least 10 characters long and follow any additional secure password rules defined by your organization.
`Enter encryption key:`	Enter a string of at least 10 characters for the encryption key. The ancryption key should contain only alphanumeric characters. Special characters should not be used. If you leave this entry blank and press Enter, the password utility will use the default key for the Spectrum installation.
`Enter installation type:`	Enter one of the following to specify the Spectrum installation type: • 1—For a server only installation • 2—For a database only installation • 3—For a standard installation (both database and server on the local host machine)
`Do you wish to change the password?`	Press `y` to change and encrypt the new password.

6. Restart the Spectrum server.

For instructions on how to restart the Spectrum server, see , see **Starting and Stopping the Server** on page 8.

The Neo4j repository password is now changed. You can view the password in its encrypted format in the `spectrum-container.properties` file. The `spectrum.repository.password` setting is located in the `Spectrum repository settings` section of this file.

4 - Data Sources

In this section

Connections.....95
Defining Connections.....95
Compression Support for Cloud File Servers.....146
Deleting a Connection.....147

Connections

A connection is a database, file server, cloud service, or other source of data that you want to process through Spectrum Technology Platform. Spectrum Technology Platform can connect to over 20 types of data sources.

To connect Spectrum Technology Platform to a data source, you need to define the connection first, before you can define the input XML. Similarly, if you want to write dataflow output to a database, you must first define the database as an external resource.

Let's say, in your organization, data resides in disparate sources, such as Salesforce, Apache Cassandra, Hadoop, Dynamo DB, SQL server, as well as CSV files.

To access your data set:

1. You need to first connect to all these data sources. Spectrum Technology Platform allows you to create connections to all these and many more data sources, which you will see in the subsequent sub-sections.
2. Once you successfully establish a connection, you can access it in:
   - Various stages of **Spectrum Enterprise Designer**, such as:
     - **Read From DB**
     - **Read From File**
     - **Read from Hadoop Sequence File**
     - **Read From Hive File**
     - **Read from HL7 File**
     - **Read from NoSQL DB**
     - **Read from SAP**
     - **Read from Spreadsheet**
     - **Read from Variable Format File**
     - **Read From XML**
   - The Catalog, Modeling, and Profiling processes in **Spectrum Discovery**

Defining Connections

To define a new connection in Spectrum Technology Platform, use one of these interfaces:

- The **Connections** page in Management Console.
- **Connect** menu option of Spectrum Discovery

Note: If you want to read from or write to data located in a file on the Spectrum Technology Platform server itself there is no need to define a connection.

Connecting to Amazon

Connecting to Amazon DynamoDB

1. Access the **Connections** page using one of these:

Spectrum Management Console:	Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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. **Note:** By default, the HTTP port is 8080. Click **Resources** > **Connections**.
Spectrum Discovery:	Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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. **Note:** By default, the HTTP port is 8080. Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, choose **Amazon DynamoDB**.
5. In the **Access Key ID** field, enter the 20-character alpha-numeric sequence provided to you to access your Amazon AWS account.
6. In the **Secret Access Key** field, enter the 40-character key needed to authenticate the connection.
7. In the **Region** field, select the region of the Amazon AWS account.
8. To test the connection, click **Test**.

9. Click **Save**.

Amazon DynamoDB Limitations

1. Hierarchical data types like lists, sets and maps are interpreted as String data types. This is because these data types are not supported.
2. Null values in a DynamoDB data source are interpreted as empty column values.
3. The `count` aggregate function is not supported in query on Model Store.

Connecting to Amazon S3

1. Access the **Connections** page using one of these:

Spectrum Management Console:	Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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. **Note:** By default, the HTTP port is 8080. Click **Resources** > **Connections**.
Spectrum Discovery:	Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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. **Note:** By default, the HTTP port is 8080. Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, choose **Cloud**.
5. In the **Cloud service** field, choose **AmazonS3**.
6. In the **Bucket name** field, enter the bucket name as defined in your Amazon S3 cloud service. This is the bucket where Spectrum Technology Platform will read and write files.
7. Enter your access key and secret key assigned to you by Amazon.
8. In the **Storage Type**, field select the level of redundancy that you want to allow for data storage.

Standard	The default level of redundancy provided by Amazon S3.

Reduced redundancy	Stores non-critical and easily-reproducible data at lower levels of redundancy. This provides fairly reliable storage at a lower cost.

9. In the **Encryption** section, select the encryption method for the data. You can select server side encryption, client side encryption, or both.

Server side key	The data is encrypted and decrypted at the server side. Your data is transmitted in plain text to the Amazon cloud service where it is encrypted and stored. On retrieval, the data is decrypted by the Amazon cloud service then transmitted in plain text to your system. You have two options for specifying the key: • **AWS managed**: The key is automatically generated by the Amazon S3 cloud service. • **Customer provided**: Enter the key to be used by the Amazon S3 cloud service to encrypt and decrypt the data on the server side.
Client side key	The data is encrypted and decrypted at the client side. The data is encrypted locally on your client system then transmitted to the Amazon S3 cloud storage. On retrieval, the data is transmitted back in an encrypted format to your system and is decrypted on the client system. **Client side key**: Enter the key to be used by your client system to encrypt and decrypt the data.

   If you select both **Server side key** and **Client side key**, encryption and decryption is performed at both server and client sides. Data is first encrypted with your client side key and transmitted in an encrypted format to Amazon, where it is again encrypted with the server side key and stored. On retrieval, Amazon first decrypts the data with the server side key, transmitting the data in an encrypted format to your system, where it is finally decrypted with the client side key.

   For more information about Amazon S3 encryption features, see:

   docs.aws.amazon.com/AmazonS3/latest/dev/UsingEncryption.html

10. If you want to set access permissions, in the **Permissions** section, click +.

    The three kinds of Grantees are:

Everyone	Every one else other than Authenticated Users and Log Delivery group.
AuthenticatedUsers	For users who are logged into Amazon.
LogDelivery	For users who write activity logs in a user-specified Bucket, if Bucket Logging is enabled.

    For each Grantee, select the desired permissions:

Open/Download	Allow the user to download the file.

View	Allow the user to view the current permissions on the file.
Edit	Allow the user to modify and set the permissions on the file.

11. To test the connection, click **Test**.
12. Click **Save**.

Connecting to Amazon SimpleDB

1. Access the **Connections** page using one of these:

Spectrum Management Console:	Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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. **Note:** By default, the HTTP port is 8080. Click **Resources** > **Connections**.
Spectrum Discovery:	Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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. **Note:** By default, the HTTP port is 8080. Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, choose **Amazon SimpleDB**.
5. In the **Access Key ID** field, enter the 20-character alpha-numeric sequence provided to you to access your Amazon AWS account.
6. In the **Secret Access Key** field, enter the 40-character key needed to authenticate the connection.
7. To test the connection, click **Test**.
8. Click **Save**.

Amazon SimpleDB Limitations

Write Limitation

In the Write to DB stage, the write mode **Update** is not available when writing to an Amazon SimpleDB table. The **Insert** option handles both insert and update operations. It differentiates between an insert and an update using the unique value of the `ItemName` column which is present in all Amazon SimpleDB tables.

Reason: An update query requires a Primary Key for each record of the table to be updated, which is not supported by Amazon SimpleDB databases.

Read Limitation

The aggregate functions `SUM` and `AVG` are not supported while executing queries on Model Store.

Connecting to Apache Cassandra

1. Access the **Connections** page using one of these:

Spectrum Management Console:	Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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. **Note:** By default, the HTTP port is 8080. Click **Resources** > **Connections**.
Spectrum Discovery:	Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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. **Note:** By default, the HTTP port is 8080. Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, choose **Apache Cassandra**.
5. In the **Host** field, enter the machine name or the IP on which the Apache Cassandra database is installed.
6. In the **Keyspace** field, enter the name of the keyspace of the data center you wish to access.
7. In the **Port** field, enter the port on which the Apache Cassandra database is configured.
8. Enter the user name and password to use to authenticate to the Cassandra database.
9. In the **Consistency Level** field, select how consistent data rows must be across replica nodes for a successful data transaction. This can be at least one, or all, or a combination of available nodes.
10. In the **Fetch Size**, enter the number of resultset rows you wish to fetch on each read transaction.
11. To test the connection, click **Test**.
12. Click **Save**.

Apache Cassandra Limitation

The `count` aggregate function is not supported in query on Model Store.

Connecting to Azure Cloud

1. Access the **Connections** page using one of these:

Spectrum Management Console:	Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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. **Note:** By default, the HTTP port is 8080. Click **Resources** > **Connections**.
Spectrum Discovery:	Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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. **Note:** By default, the HTTP port is 8080. Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, choose **Cloud**.
5. In the **Cloud service** field, choose **AzureBlobStorage**.
6. In the **Protocol** field select whether you want the connection between Azure and Spectrum Technology Platform to use HTTP or HTTPS.
7. In the **Account Name** field, enter the name of your Azure storage account.
8. In the **Access Key** field, enter the access key to your Azure account.
9. To test the cloud connection, click **Test**.
10. Click **Save**.

Connecting to a Flat File

Connecting to a Delimited Flat File

1. Access the **Connections** page using one of these:

Spectrum Management Console:	Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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. **Note:** By default, the HTTP port is 8080. Click **Resources** > **Connections**.
Spectrum Discovery:	Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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. **Note:** By default, the HTTP port is 8080. Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** box, select Flat File.
5. Enter the **File Path** by clicking the **Browse** button and locating the file.

   Note: While building a model store on a flat file connection, *periods (.)* if any, in the column name automatically gets converted to *underscores (_)*.

6. Select the **Character Encoding** of the flat file from the drop-down.
7. Select the **Record Type** as Delimited.
8. In **Field Delimiter**, select the expected separator between any two fields of a file record.

   If the delimiter does not appear on the list, click the Add field delimiter + button to define the delimiter.

9. Optional: If there is one, select the **Text Qualifier** that encloses field values.
10. In **Line Separator**, the value `Default` is selected, indicating that the expected line separator depends on whether Spectrum Technology Platform is running on a Windows system.
11. To specify whether the first row of the file is a header row, shift the **First row is header row** slider to either **ON** or **OFF**.
12. To specify whether the data type of the various fields in any record of the file should be automatically detected, shift the **Detect data type from file** slider to either **ON** or **OFF**.
13. To skip malformed records during file parsing, shift the **Skip Malformed Records** slider to **ON**.
14. Click the **Preview** button to view field values based on selections you made in previous steps of this procedure.
15. Click **Test**.
    A message confirms the successful test of the connection.
16. Click **Save**.
    A message confirms the successful creation of the connection.

In order to view a sample record fetched using the created Delimited Flat File connection, click **Preview** in the header bar. File records will be fetched and the Fields sorted according to the details provided by you.

Connecting to a Fixed-Width Flat File

1. Access the **Connections** page using one of these:

   Spectrum Management Console: Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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.

   Note: By default, the HTTP port is 8080.

   Click **Resources** > **Connections**.

Spectrum Discovery: Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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.

Note: By default, the HTTP port is 8080.

Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** box, select Flat File.
5. Enter the **File Path** by clicking **Browse** and selecting the directory of the file.
6. Select the **Character Encoding** of the flat file from the drop-down.
7. Select the **Record Type** as Fixed Width.
8. In the **Record Length** field, enter the total number of characters in a file record.
9. Click **Add Field** to add a row for a field in a file record.
10. In the **Name** column, enter the name for the field value.
11. In the **Type** column, select the data type of the field value.
12. In the **Start Position** column, enter the position in the file record at which of the field value begins.

    For the first field in a file record, the **Start Position** counting begins from 1.
13. In the **Length** field, enter the total number of characters the field covers, including the character at the **Start Position**.

    The sum of the **Start Position** and **Length** values for any field should be less than or equal to the **Record Length**

    If the File Record is:

```
01234Rob Smith29Precisely
```

    Record Length = 25

    For the field 'Name':

    Start Position = 6

    Length = 9

```
Name = Rob Smith
```

14. Check the **Trim** check box if you wish to trim any white spaces at the beginning and/or end of a field value.
15. Repeat step 10 through step 15 to add details for all fields expected in file records.
16. Click the **Preview** button to view field values based on selections you made in previous steps of this procedure.
17. Click **Test**.
    A message confirms the successful test of the connection.
18. Click **Save**.
    A message confirms the successful creation of the connection.

In order to view a sample record fetched using the created Fixed Width Flat File connection, click **Preview** in the header bar. File records will be fetched and the Fields sorted according to the details provided by you.

Date Time Formats in a File Connection

Date and time values read from files using a File Connection in Spectrum Technology Platform need to adhere to date-time formats described here.

Accepted Date Time Formats

- Date: *yyyy-mm-dd*"
- Datetime: *yyyy-mm-dd HH:mm:ss*
- Time: *HH:mm:ss*

Delimited Files

If the **Detect type** feature is turned on while configuring the Delimited File Connection, then the date and time values in the file records, which adhere to the above formats, are automatically detected as Date type.

If a date-time value does not adhere to one of the accepted formats, the value is read as a String type value instead of a Date type value.

Fixed Width Files

For Fixed Width files, date type values are configured while creating the Fixed Width File Connection. Hence these values are read as Date type values, irrespective of whether they adhere to the accepted formats or not.

If the date-time value in a Fixed Width file does not adhere to the accepted formats, it needs to be handled using **Transformations** at the logical model creation stage by applying this *Conversion* category function to the value:

```
parsedate(String date, String format)
```

In this, the *date* is the value received from the file, while the *format* is the date-time format in which the value is received from the file. This helps to parse the date-time value correctly.

For example, if the date = `23-Feb-2008`, then the format = `dd-MMM-yyyy`.

Resulting Value Formats

While previewing data in a model store:

- If the value has been read as a date/time value, it is reflected in one of the accepted date/time formats in the preview.
- If the value has been read as a String value, it is reflected as it is in the preview.

Connecting to an FTP Server

In order for Spectrum Technology Platform to access files on an FTP server you must define a connection to the FTP server using Spectrum Management Console. Once you do this, you can create dataflows in Spectrum Enterprise Designer that can read data from, and write data to, files on the FTP server.

Before connecting to an FTP server, verify that the timeout settings on your FTP server are appropriate for the jobs that will use this connection. Depending on the design of a job, there may be periods of time when the connection is idle, which may cause the connection to time out. For example, you may have a dataflow with two Read from File stages connected to an Import To Model stage. While the Import To Model stage is reading records from one Read from File stage, the other will be idle, possibly causing its connection to the FTP server to time out. Consider setting the timeout value on your FTP server to 0 to prevent connections from timing out.

> **Note:** The FTP server must be running in active connection mode. Passive connection mode is not supported.

1. Access the **Connections** page using one of these:

   Spectrum Management Console: Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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.

   > **Note:** By default, the HTTP port is 8080.

   Click **Resources** > **Connections**.

   Spectrum Discovery: Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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.

Note: By default, the HTTP port is 8080.

Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, choose **FTP**.
5. In the **User name** and **Password** fields, enter the credentials to use to authenticate to the FTP server. This is required only if the FTP server requires it.
6. In the **Host** field, enter the host name or IP address of the FTP server.
7. In the **Port** field, enter the network port number the server uses for FTP.
8. Click **Test** to verify that the Spectrum Technology Platform server can connect to the FTP server.
9. Click **Save**.

Connecting to an SFTP Server

In order for Spectrum Technology Platform to access files on an SFTP server you must define a connection to the SFTP server using Spectrum Management Console.

Before connecting to an SFTP server, verify that the timeout settings on your SFTP server are appropriate for the jobs that will use this connection. Depending on the design of a job, there may be periods of time when the connection is idle, which may cause the connection to time out. For example, you may have a dataflow with two Read from File stages connected to an Import To Model stage. While the Import To Model stage is reading records from one **Read from File** stage, the other will be idle, possibly causing its connection to the SFTP server to time out. Consider setting the timeout value on your SFTP server to 0 to prevent connections from timing out.

Note: The SFTP server must be running in active connection mode. Passive connection mode is not supported.

1. Access the **Connections** page using one of these:

   Spectrum Management Console: Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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.

   Note: By default, the HTTP port is 8080.

Click **Resources** > **Connections**.

Spectrum Discovery: Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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.

Note: By default, the HTTP port is 8080.

Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, choose **SFTP**.
5. In the **Host** field, enter the host name or IP address of the SFTP server.
6. In the **Port** field, enter the network port number that the server uses for SFTP. The default is 22.
7. Toggle the **Strict Host Key Checking** button to **Yes**, if you want to enable it. Default is **No**. If the strict host key checking is enabled, *ssh* does not automatically add host keys to the known host file. This is an additional security feature.
8. In the **Known Host File** field, browse the location of the file that maintains known hosts details, and select it.

   Note: If you disable **Strict Host Key Checking**, this field is not displayed. However, this detail is mandatory if strict host key checking is enabled.

9. Enter the **Username**.
10. Select the preferred **Authentication Type**. It can be `Password` or `Key-Based`. Default is `Password`.
    - **Password**: If you selected this as the authentication type, **Password** field is displayed below the **Authentication Type** field. Enter the required password in this field.
    - **Key-Based**: If you selected a key-based authentication, browse and select the **Private Key File**, and enter the **Passphrase** shared by the host server administrator. The pass phrase is needed only if the key is configured using this option.
11. Click **Test** to verify that the Spectrum Technology Platform server can connect to the SFTP server.
12. Click **Save**.

Connecting to Google Cloud Storage

1. Access the **Connections** page using one of these:

   Spectrum Management Console: Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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.

   Note: By default, the HTTP port is 8080.

   Click **Resources** > **Connections**.

   Spectrum Discovery: Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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.

   Note: By default, the HTTP port is 8080.

   Click **Connect**.

2. Click the Add connection button +.
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, choose **Cloud**.
5. In the **Cloud service** field, choose **GoogleCloudStorage**.
6. In the **Bucket name** field, enter the bucket name as defined in your Google cloud service. This is the bucket where Spectrum Technology Platform will read and write files.
7. Enter your the application name, service account, and private key file provided by Google.

   Note: Ensure the private key file is present on the Spectrum Technology Platform server.

8. You can set access permissions in the **Permissions** section.

Manage your data and permission	Allows the user to manage the data and permissions.
View your data	Allows the user to view data.
Manage your data	Allows the user to manage data.

9. To test the connection, click **Test**.

10. Click **Save**.

For more information, see Google's **Service Account Authentication** documentation.

Connecting to Hadoop

Connect to the Hadoop system to use the stages, such as **Read from Hadoop Sequence File**, **Write to Hadoop Sequence File**, **Read From File**, **Write to File**, **Read From XML**, **Write to XML**, **Read From Hive File**, **Write to Hive File**, and **Read from HL7 File**, in **Spectrum Enterprise Designer**.

Attention: Spectrum Technology Platform does not support *Hadoop 2.x* for Kerberos on Windows platforms.

Follow these steps to connect to the Hadoop system:

1. Access the **Connections** page using one of these:

Spectrum Management Console:	Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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. **Note:** By default, the HTTP port is 8080. Click **Resources** > **Connections**.
Spectrum Discovery:	Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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. **Note:** By default, the HTTP port is 8080. Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, choose **HDFS**
5. In the **Host** field, enter the host name or IP address of the NameNode in the HDFS cluster.
6. In the **Port** field, enter the network port number.

7. In **User**, select one of these options:

   Server user — Choose this option if authentication is enabled in your HDFS cluster. This option will use the user credentials that the Spectrum Technology Platform server runs under to authenticate to HDFS.

   User name — Choose this option if authentication is disabled in your HDFS cluster.

8. Check **Kerberos** if you wish to enable Kerberos authentication feature for this HDFS file server connection.
9. If you have opted to enable **Kerberos** authentication, then enter the path of the keytab file in the **Keytab file path** field.

   Note: Ensure the key tab file is placed on the Spectrum Technology Platform server.

10. In the **Protocol** field, select one of:

    WEBHDFS — Select this option if the HDFS cluster is running HDFS 1.0 or later. This protocol supports both read and write operations.

    HFTP — Select this option if the HDFS cluster is running a version older than HDFS 1.0, or if your organization does not allow the WEBHDFS protocol. This protocol only supports the read operation.

    HAR — Select this option to access Hadoop archive files. If you choose this option, specify the path to the archive file in the **Path** field. This protocol only supports the read operation.

11. Expand the **Advanced options**.
12. If you selected the WEBHDFS protocol, you can specify these advanced options as required:

    Replication factor — Specifies how many data nodes to replicate each block to. For example, the default setting of 3 replicates each block to three different nodes in the cluster. The maximum replication factor is 1024.

    Block size — Specifies the size of each block. HDFS breaks up a file into blocks of the size you specify here. For example, if you specify the default 64 MB, each file is broken up into 64 MB blocks. Each block is then replicated to the number of nodes in the cluster specified in the **Replication factor** field.

    File permissions — Specifies the level of access to files written to the HDFS cluster by Spectrum Technology Platform. You can specify read and write permissions for each of these options:

    Note: The *Execute* permission is not applicable to Spectrum Technology Platform.

    User — This is the user specified above, either **Server user** or the user specified in the **User name** field.

Group	This refers to any group of which the user is a member. For example, if the user is john123, then Group permissions apply to any group of which john123 is a member.
Other	This refers to any other users as well as groups of which the specified user is not a member.

13. Use the **File permissions** descriptions below to define the server properties for Hadoop to ensure that the sorting and filtering features work as desired when the connection is used in a stage or activity. To add properties, complete one of these steps:

    - Click + and add the properties and their respective values in the **Property** and **Value** fields.
    - Click and upload your configuration XML file. The XML file should be similar to `hdfs-site.xml`, `yarn-site.xml`, or `core-site.xml`.

      Note: Place the configuration file on the server.

File permissions and parameters - Hadoop 1.x

This section applies to this stage and activity:

- Stage - **Read from Sequence File**
- Activity - **Run Hadoop Pig**

fs.default.name

Specifies the node and port on which Hadoop runs. For example, `hdfs://152.144.226.224:9000`

mapred.job.tracker

Specifies the host name or IP address, and port on which the MapReduce job tracker runs. If the host name is entered as local, then jobs are run as a single map and reduce task. For example, `152.144.226.224:9001`

dfs.namenode.name.dir

Specifies where on the local files system a DFS name node should store the name table. If this is a comma-delimited list of directories, then the name table is replicated in all of the directories, for redundancy. For example, `file:/home/hduser/Data/namenode`

hadoop.tmp.dir

Specifies the base location for other temporary directories. For example, `/home/hduser/Data/tmp`

File permissions and parameters - Hadoop 2.x

This section applies to this stage and activity:

- Stage - **Read from Sequence File**
- Activity - **Run Hadoop Pig**

fs.defaultFS

Specifies the node and port on which Hadoop runs. For example, `hdfs://152.144.226.224:9000`.

NOTE: For Spectrum versions 11.0 and earlier, the parameter name `fs.defaultfs` must be used. Note the case difference. For versions 11 SP1 and later, both the names `fs.defaultfs` and `fs.defaultFS` are valid. We recommend using parameter name `fs.defaultFS` for releases 11.0 SP1 and later.

yarn.resourcemanager.resource-tracker.address

Specifies the host name or IP address of the Resource Manager. For example, `152.144.226.224:8025`

yarn.resourcemanager.scheduler.address

Specifies the address of the Scheduler Interface. For example, `152.144.226.224:8030`

yarn.resourcemanager.address

Specifies the address of the Applications Manager interface that is contained in the Resource Manager. For example, `152.144.226.224:8041`

mapreduce.jobhistory.address

Specifies the host name or IP address, and port on which the MapReduce Job History Server is running. For example, `152.144.226.224:10020`

mapreduce.application.classpath

Specifies the CLASSPATH for Map Reduce applications. This CLASSPATH denotes the location where classes related to Map Reduce applications are found. The entries should be comma separated.

For example:

```
$HADOOP_CONF_DIR, $HADOOP_COMMON_HOME/share/hadoop/common/*,
$HADOOP_COMMON_HOME/share/hadoop/common/lib/*,
$HADOOP_HDFS_HOME/share/hadoop/hdfs/*,
$HADOOP_HDFS_HOME/share/hadoop/hdfs/lib/*,
$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*,
$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib/*,
$HADOOP_YARN_HOME/share/hadoop/yarn/*,
$HADOOP_YARN_HOME/share/hadoop/yarn/lib/*
```

mapreduce.app-submission.cross-platform

Handles various platform issues that arise if your Spectrum server runs on a Windows machine, and you install Cloudera on it. If your Spectrum server and Cloudera are running on different Operating Systems, then enter the value of this parameter as `true`. Otherwise, mark it as `false`.

Note: Cloudera does not support Windows clients. Configuring this parameter is a workaround, and not a solution to all resulting platform issues.

File permissions and parameters - Kerberos

This section applies to this stage and activity:

- Stage - **Read from Sequence File**
- Activity - **Run Hadoop Pig**

If you have selected the **Kerberos** check box, add these Kerberos configuration properties:

hadoop.security.authentication

The type of authentication security being used. Enter the value `kerberos`.

yarn.resourcemanager.principal

The Kerberos principal being used for the resource manager for your Hadoop YARN resource negotiator. For example: `yarn/_HOST@HADOOP.COM`

dfs.namenode.kerberos.principal

The Kerberos principal being used for the namenode of your Hadoop Distributed File System (HDFS). For example, `hdfs/_HOST@HADOOP.COM`

dfs.datanode.kerberos.principal

The Kerberos principal being used for the data node of your Hadoop Distributed File System (HDFS). For example, `hdfs/_HOST@HADOOP.COM`

File permissions and parameters - Hadoop 1.x

This section applies to these stages:

- Stage **Read from File**
- Stage **Write to File**
- Stage **Read from Hive ORC File**
- Stage **Write to Hive ORC File**

fs.default.name

Specifies the node and port on which Hadoop runs. For example, `hdfs://152.144.226.224:9000`

File permissions and parameters - Hadoop 2.x

This section applies to these stages:

- Stage **Read or write from File**
- Stage **Read or write from Hive ORC File**

fs.defaultFS

Specifies the node and port on which Hadoop runs. For example, `hdfs://152.144.226.224:9000`

NOTE: For Spectrum versions 11.0 and earlier, the parameter name `fs.defaultfs` must be used. Note the case difference. For versions 11 SP1 and later, both the names `fs.defaultfs` and `fs.defaultFS` are valid. We recommend using parameter name `fs.defaultFS` for releases 11.0 SP1 and later.

14. To test the connection, click **Test**.
15. Click **Save**.

After you have defined a connection to an HDFS cluster, it becomes available in source and sink stages in Spectrum Enterprise Designer, such as Read from File and Write to File. You can select the HDFS cluster when you click **Remote Machine** when defining a file in a source or sink stage.

Compression Support for Hadoop

Spectrum Technology Platform supports the compression formats `gzip (.gz)` and `bzip2 (.bz2)` on Hadoop. While using the **Read from File** and **Write to File** stages with an HDFS connection, include the extension corresponding to the required compression format (`.gz` or `.bz2`) in the **File name** field. The file is decompressed or compressed based on the specified compression extension. Spectrum Technology Platform handles the compression and decompression of the files.

Connecting to Hive

You can connect to Hive databases through the driver provided by Spectrum Technology Platform or by adding the Apache JDBC driver. The driver provided by Spectrum Technology Platform (*hive-jdbc-1.2.2-batch-18.2.jar*) is an extended version of Apache Hive driver and supports batch processing. It is located here:

`SpectrumDirectory``\server\modules\bigdata\drivers\hive`, where *SpectrumDirectory* is the folder where you have installed the Spectrum Technology Platform server.

For information about manually adding a JDBC driver, see **Manually Adding a JDBC Driver** on page 117.

Note: The class path for the Hive JDBC driver is:
`com.pb.spectrum.hive.jdbc.batch.HiveBatchDriver`

Connecting to a JDBC Database

Define a connection using the **Connections** page. You can go to this page through **Spectrum Management Console** or through **Spectrum Discovery**.

1. Access the **Connections** page using one of these:

Spectrum Management Console:	Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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. **Note:** By default, the HTTP port is 8080. Click **Resources** > **Connections**.
Spectrum Discovery:	Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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. **Note:** By default, the HTTP port is 8080. Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, select type of database you want to connect to.

   Spectrum Data Federation includes JDBC drivers for SQL Server, Oracle, and PostgreSQL databases. If you want to connect to a different database type, you must add the JDBC driver before defining a connection.

5. In the **URL** field, enter the JDBC connection URL. Your database administrator can provide this URL.

   For example, to connect to a MySQL database named "SampleDatabase" hosted on a server named "MyServer" you would enter:

   ```
   jdbc:mysql://MyServer/SampleDatabase
   ```

6. There may be additional fields you need to fill in depending on the JDBC driver. The fields represent properties in the connection string for the JDBC driver you selected in the **Type** field. See the JDBC driver provider's documentation or your database administrator for information about the specific connection properties and values required by the connection type.
7. Click **Save**.
8. Test the connection by checking the box next to the new connection and clicking the **Test** button.

Importing a JDBC Driver

Spectrum Technology Platform can access data from any database using a JDBC driver. Drivers for SQL, Oracle, and PostgreSQL are provided with the Spectrum Data Federation, which also includes drivers for other types of databases. If Spectrum Technology Platform does not come with a driver for the type of database you need, you can add a JDBC driver.

In this procedure you will import a JDBC driver by copying the driver files to the Spectrum Technology Platform server. When you complete this procedure the driver will be available to use when defining a JDBC database connection in Spectrum Management Console.

> **Note:** This procedure works for *JDBC 4.x* drivers. If the driver you want to add uses an older version of JDBC you must add the driver manually in Spectrum Management Console. For more information, see **Manually Adding a JDBC Driver** on page 117

1. Put all the JDBC driver files for the database into a folder named:

   `Name.jdbc`

   Where *Name* is any name you want. The folder name must end with `.jdbc`.

2. Log in to the server running Spectrum Technology Platform.
3. Copy the folder containing the driver to this folder:

   `SpectrumDirectory\server\drivers`

   The driver is automatically imported.

4. To verify that the driver was successfully imported, log in to Spectrum Management Console and go to **System** > **Drivers**. The driver should be listed.

   If the driver is not listed, open the System Log in Spectrum Management Console and look for errors related to deploying JDBC drivers.

Manually Adding a JDBC Driver

Spectrum Technology Platform can access data from any database using a JDBC driver. Drivers for SQL, Oracle, and PostgreSQL are provided with the Spectrum Data Federation, which also includes drivers for other types of databases. If Spectrum Technology Platform does not come with a driver for the type of database you need, you can add a JDBC driver.

In this procedure you will add JDBC driver files to the server then manually define the connection string and connection properties. Before you begin, be sure that you understand the connection string format and properties required by the driver. You must define these accurately in order for the driver to function. You can typically find information about a driver's connection string and properties from the driver provider's website.

Note: We recommend that you use this procedure only when adding a JDBC driver that uses *JDBC 1.x*, *2.x*, or *3.x.* If the driver uses *JDBC 4.x*, we recommend that you use the import method to add the driver. For more information, see **Importing a JDBC Driver** on page 117.

1. Open Spectrum Management Console.
2. Go to **System** > **Drivers**.
3. Click the Add button .
4. In the **Name** field, enter a name for the driver. The name can be anything you choose.
5. In the **JDBC driver class name** field, enter the Java class name of the driver. You can typically find the class name in your JDBC driver's documentation.

   For example, to use the Microsoft JDBC driver, you might enter the following:

   ```
   com.microsoft.sqlserver.jdbc.SQLServerDriver
   ```

6. In the **Connection string template** field, enter the JDBC connection URL to use to connect to the database, including any properties you want to set in the connection string. Different database vendors use different connection strings so check your database's documentation for more information about the connection string.

   If the driver will be used by more than one database connection, consider using property tokens in the connection string instead of hard-coding property values that may be different for each connection. For example, if you want to have some connections use encryption and others not, you may want to define a property token for the encryption property.

   To use a property token in the connection string, use this syntax:

   ```
   ${PropertyToken}
   ```

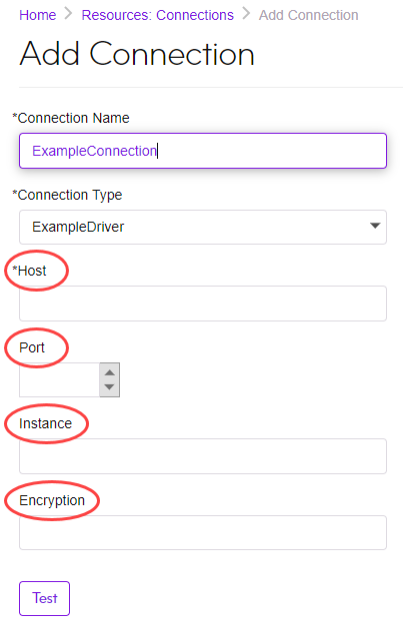
   Any property tokens you include in the connection string template will be required fields when defining a database connection.

   Note: Use the property token name `${password}` for the property that will contain the database password. By using this token name, the password will be masked in the field in Spectrum Management Console and will be encrypted in the database.

   For example, this connection string for SQL contains property tokens for host, port, instance, and encryption:

   ```
   jdbc:sqlserver://${host}:${port};databaseName=${instance};encrypt=${encryption};
   TrustServerCertificate=true
   ```

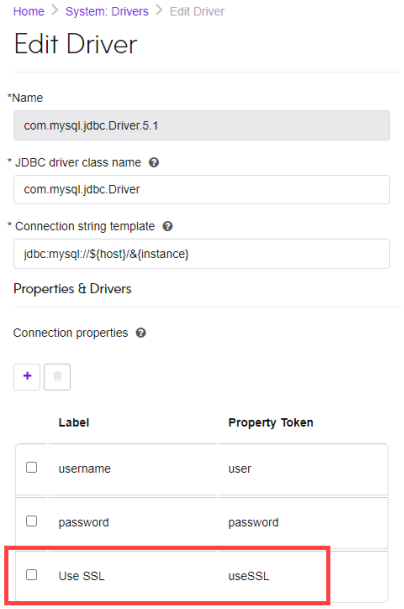
   These tokens are required fields when defining a database connection that uses this driver:



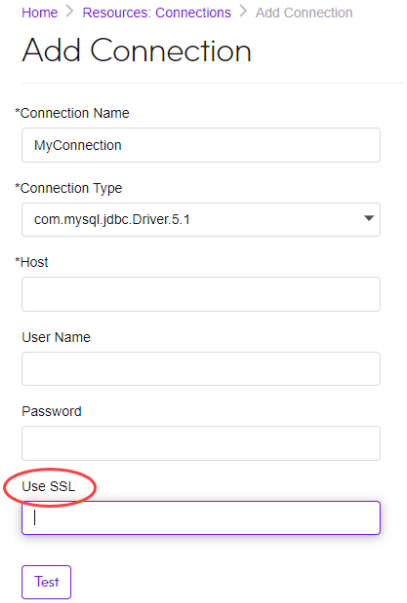
7. If there are properties that you want to make optional for database connections, define them in the **Connection Properties** section.
   a) In the **Connection properties** section, click the Add button .
   b) In the **Label** field, enter a user-friendly description of the property. The label you enter here is used as the field label in the connections window when creating a connection using this driver.
   c) In the **Property token** field, enter the token for the optional property. See the database driver's documentation for the properties supported by the driver.

      Note: Use the property token name `password` for the property that will contain the database password. By using this token name, the password will be masked in the field in Spectrum Management Console and will be encrypted in the database.

   For example, if you want to make encryption optional for database connections that use this driver, you could define the encryption property like this:



When a database connection uses this driver, the encryption property would be displayed as an optional property in the database connection:



8. Log in to the server running Spectrum Technology Platform and place the database driver file in a folder on the server. The location does not matter.
9. In the **Driver files** section, click the Add button +.
10. In the **File path** field, enter the path to the database driver file on the server.
11. Click **Save**.

Deleting an Imported JDBC Driver

JDBC drivers cannot be deleted using Spectrum Management Console if the JDBC driver was imported to Spectrum Technology Platform rather than being added manually in Spectrum Management Console. Instead, follow this procedure to delete the driver.

Important: Before deleting a driver, verify that there are no database connections using the driver.

1. Stop the Spectrum Technology Platform server.
2. Go to this folder:

   SpectrumDirectory\server\drivers

3. In the `drivers` folder, delete folder containing the driver.
4. Start the Spectrum Technology Platform server.
5. To verify that the driver has been deleted, log in to Spectrum Management Console, go to **System** > **Drivers**, and verify that the driver is no longer listed.

Supported Database Data Types

Spectrum Technology Platform supports these data types commonly used in databases:

bigdecimal	A numeric data type that supports 38 decimal points of precision. Use this data type for data that will be used in mathematical calculations requiring a high degree of precision, especially those involving financial data. The bigdecimal data type supports more precise calculations than the double data type.
boolean	A logical type with two values: true and false.
date	A data type that contains a month, day, and year. For example, 2012-01-30 or January 30, 2012. You can specify a default date format in Spectrum Management Console.
datetime	A data type that contains a month, day, year, and hours, minutes, and seconds. For example, 2012/01/30 6:15:00 PM.
double	A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52})\times2^{1023}$. In E notation, the range of values is -1.79769313486232E+308 to 1.79769313486232E+308.
float	A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{23})\times2^{127}$. In E notation, the range of values -3.402823E+38 to 3.402823E+38.
integer	A numeric data type that contains both negative and positive whole numbers between -2^{31} (-2,147,483,648) and $2^{31}-1$ (2,147,483,647).
long	A numeric data type that contains both negative and positive whole numbers between -2^{63} (-9,223,372,036,854,775,808) and $2^{63}-1$ (9,223,372,036,854,775,807).
string	A sequence of characters.
time	A data type that contains the time of day. For example, 21:15:59 or 9:15:59 PM.

Raw An Oracle data type for storing variable length binary data. Maximum size is 2000 bytes (the maximum length in Oracle 7 was 255 bytes).

Other database data types are automatically mapped to one of the supported data types as follows:

Database Data Type	Supported Data Type
Date/Time Types	
TIMESTAMP	datetime
String Types	
CHAR	string
CLOB	string
LONGVARCHAR	string
NCHAR	string
NVARCHAR	string
VARCHAR	string
Numeric Types	
BIGINT	long
DECIMAL	double
FLOAT	double
NUMERIC	bigdecimal
REAL	float
SMALLINT	integer
TINYINT	integer

Database Data Type	Supported Data Type
Boolean Types	
BIT	boolean

Supported Database Data Types for Spectrum Spatial

These database data types are automatically mapped to one of the supported data types for Spectrum Spatial.

Database Data Type	Supported Data Type
SQL Server	
tinyint	SHORT_INTEGER
smallint	SHORT_INTEGER
int	INTEGER
bigint	LONG_INTEGER
float	DOUBLE
real	DOUBLE
decimal(10, 5)	DOUBLE
numeric(10, 5)	DOUBLE
date	DATE
time	TIME
datetime	DATE_TIME
smalldatetime	DATE_TIME
char(10)	STRING

Database Data Type	Supported Data Type
varchar(10)	STRING
nchar(10)	STRING
nvarchar(10)G	STRING
binary(10)	BINARY
varbinary(10)	BINARY
PostGIS	
smallint	SHORT_INTEGER
integer	INTEGER
bigint	LONG_INTEGER
numeric(10, 5)	DOUBLE
real	DOUBLE
double precision	DOUBLE
serial	INTEGER
bigserial	LONG_INTEGER
bytea	BINARY
date	DATE
time	TIME
timestamp	DATE_TIME
character(10)	STRING

Database Data Type	Supported Data Type
character varying(10)	STRING
nchar(10)	STRING
Oracle	
NUMBER	DOUBLE
CHAR(10)	STRING
VARCHAR(10)	STRING
VARCHAR2(10)	STRING
NCHAR(10)	STRING
NVARCHAR2(10)	STRING
DATE	DATE_TIME
TIMESTAMP	DATE_TIME
BLOB	BINARY

JDBC Database connector limitations

- MongoDB/Cassandra connectors are not supported via PrestoDB in Spectrum Discovery. There are separate connectors for MongoDB and Cassandra.
- `Write to Any DB` via Presto is not recommended by Presto DB and so is not supported by Presto JDBC connector.

Connecting to Knox

An Apache Knox Gateway allows you to access a Hadoop service through the Knox security layer.

With this connection, you can create flows in the Spectrum Enterprise Designer using stages in Enterprise Big Data to read data from and write data to Hadoop via Knox.

1. Access the **Connections** page using one of these:

   Spectrum Management Console: Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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.

   Note: By default, the HTTP port is 8080.

   Click **Resources** > **Connections**.

   Spectrum Discovery: Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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.

   Note: By default, the HTTP port is 8080.

   Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, choose `Gateway`.
5. In the **Gateway Type** field, choose **Knox**.
6. In the **Host** field, enter the host name or IP address of the node in the HDFS cluster running the gateway.
7. In the **Port** field, enter the port number for the Knox gateway.
8. In the **User Name** field, enter the user name for the Knox gateway.
9. In the **Password** field, enter the password to authorize you access to the Knox gateway.
10. In the **Gateway Name** field, enter the name of the Knox gateway you wish to access.
11. In the **Cluster Name** field, enter the name of the Hadoop cluster to be accessed.
12. In the **Protocol** field, choose `webhdfs`.
13. In the **Service Name** field, enter the name of the Hadoop service to be accessed.
14. To test the connection, click **Test**.
15. Click **Save**.

After you have defined a Knox connection to an HDFS cluster, the connection can be used in Spectrum Enterprise Designer, in the stages **Read from File** and **Write to File**. You can select the HDFS cluster when you click **Remote Machine** when defining a file in a source or sink stage.

Connecting to a Windows Mapped Drive

When Spectrum Technology Platform is running on a Windows server, it can access data on the server's mapped drives. Since the Spectrum Technology Platform server runs as a Windows service under a particular user account (often the Local System account) you need to define the mapped drive in the server's start-up process in order for it to be visible in Spectrum Enterprise Designer and Spectrum Management Console.

1. Stop the Spectrum Technology Platform server.
2. Under the folder where the Spectrum Technology Platform server is installed, go to `server\bin\wrapper`. For example, `C:\Program Files\Precisely\Spectrum\server\bin\wrapper`.
3. Open the file `wrapper.conf` in a text editor.

   Important: In the following steps you will add new properties to this file. It is important that you follow these instructions precisely and only add and modify the properties described in the following steps. Do not modify any of the other properties in this file.

4. Add these lines:

```
wrapper.share.1.location
wrapper.share.1.target
wrapper.share.1.type
wrapper.share.1.account
wrapper.share.1.password
```

5. In the `wrapper.share.1.location` property, specify the location of the mapped drive in UNC format.

   Note: Do not include a trailing backslash in the UNC.

   For example,

```
wrapper.share.1.location=\\myserver\share
```

6. In the `wrapper.share.1.target` property, specify the drive letter to assign to this mapped drive.

   For example,

```
wrapper.share.1.target=Y:
```

7. In the `type` property, specify `DISK`.

   For example,

```
wrapper.share.1.type=DISK
```

8. If the share you are connecting to requires a user name and password, specify the user name in the `wrapper.share.1.account` property and specify the password in the `wrapper.share.1.password` property.

   For example,

```
wrapper.share.1.account=domain\user123
wrapper.share.1.password=mypassword1
```

   Note: If the Spectrum Technology Platform server service is running under the Local System user, you cannot specify a user name and password. If the share requires a user name and password you must modify the service to run under a different account.

Example

This example shows two mapped drives being defined in the `wrapper.conf` file.

```
wrapper.share.1.location=\\myserver\data
wrapper.share.1.target=Y:
wrapper.share.1.type=DISK
wrapper.share.1.account=sample\user
wrapper.share.1.password=samplepass
wrapper.share.2.location=\\myserver\moredata
wrapper.share.2.target=Z:
wrapper.share.2.type=DISK
wrapper.share.2.account=sample\user
wrapper.share.2.password=samplepass
```

Connecting to Marketo

1. Access the **Connections** page using one of these:

Spectrum Management Console:	Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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. **Note:** By default, the HTTP port is 8080.

Click **Resources** > **Connections**.

Spectrum Discovery: Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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.

Note: By default, the HTTP port is 8080.

Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, choose **Marketo**.
5. In the **Endpoint URL** field, enter the endpoint URL of your Marketo account.

   To find your endpoint URL, log in to your Marketo account and go to **Admin** > **Integration** > **Web Services**. The endpoint URL is under the heading **REST API** and is in this format:

   `https://AccountID.mktorest.com/rest`

   Copy the portion of the URL before `/rest`. For example, `https://AccountID.mktorest.com`.

6. Enter the client ID and secret key for your Marketo account.

   To find your client ID and secret key, log in to your Marketo account and go to **Admin** > **Integration** > **LaunchPoint** > **API Rest** > **View Details**. The pop-up window displays the details.

7. To test the connection, click **Test**.
8. Click **Save**.

Marketo Limitations

1. This query applies only to `List` and `Activity_type` entities. For others, provide the filter type.

```
Select * from Marketo_Table
```

2. Does not support joins except between `Lead` and `Lead_List` entities. The join query between `Lead` and `Lead_List` for a List_Id is as follows:

```
Select Lead.* from Lead Inner Join Lead_List
On Lead.ID = Lead_List.Lead_ID
And Lead_List.List_ID = <List ID>
```

Supported Entities and Operations

The entities are of these types:

1. Entity
2. Entity Update: This is a virtual table used for Update on Lead entity. For example, **Merge_Leads** should be used for merging different Marketo Leads.

Connecting to Microsoft Dynamics 365

Connecting to Microsoft Dynamics 365 Online

Spectrum Technology Platform supports connecting to Microsoft Dynamics 365 (version 9) only.

1. Access the **Connections** page using one of these:

Spectrum Management Console:	Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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. **Note:** By default, the HTTP port is 8080. Click **Resources** > **Connections**.
Spectrum Discovery:	Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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. **Note:** By default, the HTTP port is 8080. Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, choose **Microsoft Dynamics 365**.
5. In the **Deployment Type** field, select **Online**.
6. In the **Username** field, enter your Microsoft Dynamics user name.

7. In the **Password** field, enter your Microsoft Dynamics password.
8. In the **Organization Name** field, enter your organization unique name, which identifies your CRM instance.

   To find your organization unique name, log in to Microsoft Dynamics and go to **Settings** > **Customization** > **Customizations** > **Developer Resources**. Your organization unique name is displayed.
9. In the **Region** field, select the geographical region of your Microsoft Dynamics account.
10. To test the connection, click **Test**.
11. Click **Save**.

Connecting to Microsoft Dynamics 365 On Premise

This connector from Spectrum Technology Platform supports claims based authentications for Microsoft Dynamics 365 On Premises.

Prerequisites:

Import certificate to the keystore file: To import the Dynamics CRM server certificates to the Spectrum Java distribution Keystore, perform these tasks:

1. Copy the server certificates to a local folder
2. Browse this path to the Spectrum JAVA distribution: `*SpectrumDirectory*\*JavaLocation*\jre\lib\security`
3. Run this command to import the certificates:
   - Windows - `keytool -importcert -alias *certificatealiasname* -file "*certificatepath*\*certificatename*" -keystore keystore.jks`
   - Linux - `keytool -import -alias *certificatealiasname* -file "*certificatepath*/*certificatename*" -keystore keystore.jks`.

Defining Microsoft Dynamics 365 On Premise connection

Perform these steps to define a **Microsoft Dynamics 365 On Premises** connection:

1. Access the **Connections** page using one of these:

Spectrum Management Console:	Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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. **Note:** By default, the HTTP port is 8080. Click **Resources** > **Connections**.

Spectrum Discovery: Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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.

Note: By default, the HTTP port is 8080.

Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** box, lick Microsoft Dynamics 365.
5. In the **Deployment Type** box, click **On Premise**.
6. Enter your Microsoft Dynamics user name in the **Username**.
7. Enter your Microsoft Dynamics password in the **Password**.
8. Enter the name of the host in the **Host Name**.
9. Enter the name of the port In the **Port Name**.
10. Enter the URL of the STS in the **STS URL**.
11. Click **Test** to test the connection.
12. Click **Save**.

Limitations

Create/Update: Create/Update can fail if a column in an Entity is mapped to multiple Reference Entities. For example, `ParentCustomerId` in Customer can be associated to Account, Lead, and others. To resolve this, the data for this column needs to be in the following format: `ReferenceEntityName:GUID` in place of `GUID`.

Supported Entities and Operations

The entities are of these types:

- User-owned
- Organization-owned
- Business-owned
- None

Connecting to a Model Store

Connect to a model store to use the data federated from various sources such as databases, file servers, and cloud services. Once a connection is defined, you can use the data in the logical and physical models of a model store (created and deployed in Spectrum Discovery) in the **Read from DB** and **Write to DB** stages of Spectrum Enterprise Designer.

1. Access the **Connections** page using one of these:

Spectrum Management Console:	Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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. **Note:** By default, the HTTP port is 8080. Click **Resources** > **Connections**.
Spectrum Discovery:	Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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. **Note:** By default, the HTTP port is 8080. Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, choose **Model Store**.
5. In the **Model store** field, enter the name of the model store that you are establishing connection with.

   To find the names of the available model stores, access **Spectrum Discovery**, go to **Model**, and click the **Model Store** tab.

6. To test the connection, click **Test**.
7. Click **Save**.

Note: Using the **Write to DB** stage over a Model Store connection has certain limitations, such as *Create Table*, *Truncate table before inserting data*, and *Drop and recreate the table if it already exists* not being supported.

Connecting to NetSuite

While reading from and writing to a NetSuite connection, both interactive and batch modes are supported. Spectrum Technology Platform supports these NetSuite entity types:

- Standard records
- Custom records
- Saved searches
- Joins between Standard records

To connect to NetSuite:

1. Access the **Connections** page using one of these:

   Spectrum Management Console: Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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.

   Note: By default, the HTTP port is 8080.

   Click **Resources** > **Connections**.

   Spectrum Discovery: Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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.

   Note: By default, the HTTP port is 8080.

   Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, choose **NetSuite**.

5. In the **Email** field, enter the email addressed linked to the NetSuite account to be used for the connection.
6. In the **Password** field, enter the password of the NetSuite account.
7. In the **Account** field, enter the user name for the NetSuite account.
8. In the **Role** field, select the appropriate role for this connection from the roles mapped to the particular NetSuite user account.

   The **Role** field is optional. If you leave the **Role** field blank, the default role is used to log in through the connection.

   Attention: Only standard roles are supported. Custom roles are not supported.
9. To test the connection, click **Test**.
10. Click **Save**.

    Note: To `INSERT` a record using a NetSuite connection, use an `UPSERT` query with the primary key (`internalId`) blank.

NetSuite Limitations

- When querying using joins, you must cite specific columns. For example, this query is not supported:

```
select * from CUSTOMER_M
```

- Simultaneous connections to NetSuite are not supported because NetSuite allows only a single sign in for one account.
- You can only write Standard and Custom records.
- For both `UPDATE` and `UPSERT` queries, an `UPSERT` operation is performed.
- In the Write to DB stage, the maximum batch size permitted for an `insert` operation is 200 and the maximum batch size for an `update` operation is 100.

Supported Entities and Operations

The entities are of these types:

- Standard records
- Custom records
- Joins
- Saved searches

Note: In a NetSuite connection table, the primary key column is `internalId`.

Connecting to NoSQL

The supported NoSQL database types are:

- Couchbase
- MongoDB

Connect to the Hadoop system to use the stages, such as **Read from Hadoop Sequence File**, **Write to Hadoop Sequence File**, **Read From File**, **Write to File**, **Read From XML**, **Write to XML**, **Read From Hive File**, **Write to Hive File**, and **Read from HL7 File**, in **Spectrum Enterprise Designer**.

Attention: Spectrum Technology Platform does not support *Hadoop 2.x* for Kerberos on Windows platforms.

- Query NoSQL DB
- Read from NoSQL DB
- Write to NoSQL DB

1. Access the **Connections** page using one of these:

   Spectrum Management Console: Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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.

   Note: By default, the HTTP port is 8080.

   Click **Resources** > **Connections**.

   Spectrum Discovery: Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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.

   Note: By default, the HTTP port is 8080.

   Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, select any one of:

- Couchbase
- MongoDB

Note: The MongoDB connection is available based on your current Spectrum version.

Spectrum version	Spectrum library	Supported MongoDB version
Spectrum 22.1	3.12.10	Upto 4.2

Please refer **MongoDB compatibility** for details.

5. Specify the **Host**, **Port**, **Database**, **Username** and **Password** of the specific NoSQL database you wish to access.
6. Click **Test** to check that the connection to the database is successful.
7. Click **OK**.

Connecting to Salesforce

1. Access the **Connections** page using one of these:

Spectrum Management Console:	Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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. **Note:** By default, the HTTP port is 8080. Click **Resources** > **Connections**.
Spectrum Discovery:	Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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. **Note:** By default, the HTTP port is 8080. Click **Connect**.

2. Click the Add connection button +.
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, choose **Salesforce**.
5. In the **Username** field, enter the email ID registered on the Salesforce data store.
6. In the **Password** field, enter a combination of the Salesforce portal password and the security token generated through the Salesforce portal.

   For example, if your password is Sales@Test, and the security token provided to you by Salesforce is 56709367, then the Password to authenticate this Salesforce connection would be Sales@Test56709367.

7. Set the **Use default endpoint** toggle to `No` if you want to use a specific endpoint URL to access the Salesforce data. Enter the required URL in the **Salesforce URL** field displayed just below the **Use default endpoint** toggle.

   Note: The **Salesforce URL** is mandatory.

8. Set the **Use bulk read** toggle to `Yes` if you want to fetch bulk data from Salesforce. Default is `No`.

   Note: Query for bulk data fetching does not work for address and geolocation compound fields. For more information about the considerations and limitations of bulk query, see **Compound Field Considerations and Limitations** and **Bulk API Limits**.

9. To test the connection, click **Test**.
10. Click **Save**.

    Note: Audit fields are enabled on all tables by default. The Salesforce audit fields are:

    - created date
    - last modified date
    - created by
    - last modified by

Attention: Physical model created in Spectrum Technology Platform version 10 and earlier using Salesforce connections need to be opened and saved again in order to enable audit fields on their tables.

Salesforce Limitation

The `Aggregate functions` are not supported while executing queries on Model Store.

Connecting to SAP NetWeaver

Creating a SAP NetWeaver connection in Management Console using OData Services allows you to read, write and synchronize your CRM and ERP data. While reading from and writing to a SAP connection, both interactive and batch modes are supported.

To define a SAP NetWeaver connection, perform these steps:

1. Access the **Connections** page using one of these:

   Spectrum Management Console: Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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.

   Note: By default, the HTTP port is 8080.

   Click **Resources** > **Connections**.

   Spectrum Discovery: Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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.

   Note: By default, the HTTP port is 8080.

   Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, choose **SAP**.
5. In the **Username** field, enter the user name to access the SAP web service.
6. In the **Password** field, enter the password of the SAP web service.
7. In the **OdataURL** field, enter the address of the Odata web service to be used for this connection.
8. Click **Test**.
   A message confirms the successful test of the connection.
9. Click **Save**.
   A message confirms the successful creation of the connection.

Note: To perform fetch operations, an OData service must support the `$skip` and `$top` operations. If the service does not support these operations, the fetched records show inconsistencies in the model store preview.

SAP NetWeaver Limitations

For both `UPDATE` and `UPSERT` operations, an `UPDATE` operation is performed.

Supported Entities and Operations

The entity columns are of two types:

- Native: Columns with native data types are displayed with their respective data types
- Custom-defined: Columns with custom-defined data types are displayed with a blank data type

To deploy a model store derived from an SAP connection, ensure its logical and physical models include only such entities whose columns are of native data types. If the models have entities of custom-defined data types, the model store cannot be deployed.

Connecting to SharePoint

1. Access the **Connections** page using one of these:

   Spectrum Management Console: Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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.

   Note: By default, the HTTP port is 8080.

   Click **Resources** > **Connections**.

   Spectrum Discovery: Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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.

   Note: By default, the HTTP port is 8080.

   Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, choose **Cloud**.
5. In the **Cloud service** field, choose **Sharepoint**.
6. In the **Version** field, select **v2010**. Spectrum Technology Platform currently supports SharePoint version 2010.
7. In the **Protocol** field, select the protocol required to connect SharePoint.
8. In the **Server address** field, enter the host name or IP address of the SharePoint server you want to connect to.
9. Enter the user name and password to use to authenticate to SharePoint.
10. In the **Project** field, enter the specific project whose SharePoint location you want to access.
11. To test the connection, click **Test**.
12. Click **Save**.

Example

For example, say you want to create a connection to this SharePoint URL:

`https://sharepoint.example.com/sites/myportal`

You would fill in the **Protocol**, **Server address**, and **Project** fields as follows:

- **Protocol**: `https`
- **Server address**: `sharepoint.example.com`
- **Project**: `myportal`

Connecting to Splunk

1. Access the **Connections** page using one of these:

Spectrum Management Console:	Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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. **Note:** By default, the HTTP port is 8080. Click **Resources** > **Connections**.
Spectrum Discovery:	Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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.

Note: By default, the HTTP port is 8080.

Click **Connect**.

2. Click the Add connection button +.
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, choose **Splunk**.
5. In the **Username** field, enter the Splunk account user name to authenticate the Splunk instance.
6. In the **Password** field, enter the password of the Splunk account.
7. In the **Host** field, enter the address or host name of the server on which the Splunk data source is hosted.
8. In the **Port** field, enter the port number of the Splunk data source.
9. To test the connection, click **Test**.
10. Click **Save**.

Splunk Limitations

This query is not supported:

```
select count(*) from SplunkTable
```

Supported Entities and Operations

Supported Operations

`LIKE`, `ORDER BY`, `LIMIT`, `IN`, `BETWEEN`, `!=`, `<=`, `>=`, `<,>`, multiple AND/OR operators.

Supported Functions

- String Functions: `upper`, `lower`, `length`, `len`, `ltrim`, `rtrim`, `substring`, `max`, `min`
- Mathematical Functions: `abs`, `ceil`, `exp`, `floor`, `sqrt`, `round`

Note: For all other query operations, use the Splunk `search` column as explained below.

Spectrum Technology Platform provides a column `search` in the Splunk table using which you can look up the required data in the Splunk connection.

While executing a `select` query on the `SplunkTable`, use the `search` column in the `where` clause in any of these scenarios:

1. To include such search criteria which cannot be specified using ANSI SQL syntax.
2. To include such Splunk-specific search criteria which cannot be included as part of the main SQL query.

For example, this query looks for such a `_raw` value which contains the key `opp` with the value `ACC`.

```
select "_raw" from SplunkTable where "search"='search opp=ACC'
```

Connecting to Oracle Eloqua

1. Access the **Connections** page using one of these:

Spectrum Management Console:	Access Spectrum Management Console using the URL: http://*server*:*port*/management console, 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. **Note:** By default, the HTTP port is 8080. Click **Resources** > **Connections**.
Spectrum Discovery:	Access Spectrum Discovery using the URL: http://*server*:*port*/discovery, 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. **Note:** By default, the HTTP port is 8080. Click **Connect**.

2. Click the Add connection button .
3. In the **Connection Name** box, enter a name for the connection. The name can be anything you choose.

   Note: Once you save a connection you cannot change the name.

4. In the **Connection Type** field, click **Oracle Eloqua**.
5. Enter the **Site Name** which is same as Company Name.
6. Enter the user name in the **Username** field.
7. Enter the password in the **Password** field.
8. Click **Test** to test the connection..

9. Click **Save**.

Special Operations

- Use this join query to fetch contacts in a Contact List:

```
select * from Contacts inner join ContactListMembers on
Contacts.Eloqua_Contact_ID = ContactListMembers.Contact_Id where
ContactListMembers.ContactList_Id = '<id>'
```

  Use this join query to fetch contacts in a Contact Segment:

```
select * from Contacts inner join ContactSegmentMembers on
Contacts.Eloqua_Contact_ID = ContactSegmentMembers.Contact_Id where
ContactSegmentMembers.Contactlist_Id = '<id>'
```

- Use this statement to insert contacts in a Contact List:

```
insert into ContactListMembers (ContactList_ID,Contact_ID) values
('<contactlist_id>','<contact_id>')
```

- Use this statement to delete contacts from a Contact List:

```
delete from ContactListMembers where ContactList_ID = '<contactlist_id>'
 and Contact_ID = '<contact_id>'
```

Limitations

- **Create/Update:**
  - `Insert/Upsert` fails if Not Null columns are blank or do not exist
  - `Insert/Upsert` fails if values of `Unique` columns are not unique for a particular batch
  - In order to avoid a rollback exception, keep the value of **Batch count to commit** to 1
- **Read:** For custom entities, `Select` operation is only applicable on joins with Contacts entity
- **Filter:**
  - Supported filters are `=`, `!=`, `>`, `<`, `>=`, `<=`
  - There is no support for `IN` and `NOT IN` condition operators when providing more than one values
  - There is no support for `Joins` between entities
  - There is support for `AND` and `OR` operation conditional operators only for Accounts and Contacts entities
  - There is support for only `AND` conditional operator. It does not work for rest of the entities on the column
  - `=` filter does not always work on fields having `timestamp` data type

Supported Entities and Operations

These entities are supported:

- **Entity:** Denotes a table representing business entity.
- **Activity:** Denotes a table representing a business entity where data is generated based on some activity.
- **Custom Entity:** Denotes entities that are used as part of special operations provided with the Connector.

This table lists the entities and the operations supported for these.

Entity Name	Create	Read	Update	Delete	Batch Support	Maximum Batch Size
Accounts	X	X	X	X	Insert/Update*	1000
Account Groups		X				
Campaign		X				
Contacts	X	X	X	X	Insert/Update*	1000
Contact List	X	X	X	X		
Contact Segment	X	X	X	X		
Emails		X				
Email Folders		X				
Email Groups		X				
Microsites		X				
Users		X				
Visitors		X				
Activity						
Email Open		X				

Entity Name	Create	Read	Update	Delete	Batch Support	Maximum Batch Size
EmailClickthrough		X				
Email Send		X				
Subscribe		X				
Unsubscribe		X				
Bounceback		X				
WebVisit		X				
PageView		X				
FormSubmit		X				
Custom Entities						
ContactListMembers	X	X		X	Insert/Delete	1000
ContactSegmentMembers		X				

***** Update operation works as Insert.

Compression Support for Cloud File Servers

The file servers Amazon S3, Google cloud storage, and MS Azure Blobstore support the compressed formats `gzip (.gz)` and `zip (.zip)`.

The Spectrum Technology Platform handles the compression and decompression of the files written to and read from the file servers.

> **Note:** You can use the same file server to handle both normal reads and writes of files as well as compression and decompression of files.

Reading a Compressed Format File

While reading a file from the server, its compression format is derived from the metadata key property `Content-Encoding` received from the server.

Writing a Compressed Format File

While writing a file to a server, mention the required compression format: `.gz` or `.zip`. The file is compressed based on the specified compression extension.

The metadata key property `Content-Encoding` is also set according to the selected compression format. This property value is passed to the cloud file server while writing the file to it.

Deleting a Connection

You can delete the connection using any of these modules:

- Spectrum Management Console
- Spectrum Discovery

1. Access the **Connections** page of the required process.
   - In Spectrum Management Console, click **Resources** > **Connections**.
   - In Spectrum Discovery, click **Connect**.
2. Check the box next to the connection you want to delete, then click the Delete connection button .

5 - Spectrum Databases

In this section

Introduction to Spectrum Databases.....149
Installing a Spectrum Database.....149
Adding a Spectrum Database.....150
Database Pool Size and Runtime Instances.....150
Configuring database properties.....153
Deleting a Spectrum Database.....155

Introduction to Spectrum Databases

Spectrum databases contain reference data from trusted data providers that is used to enhance and validate your data. For example, to perform address validation, Spectrum Technology Platform uses official address data from postal authorities to compare your address to addresses of record. Other types of processing that use Spectrum databases include geocoding, routing, and tax jurisdiction assignment for a given address.

We update Spectrum databases periodically to provide you with the most up-to-date data from third-party data providers. Database updates occur independently from software updates, in some cases quarterly or even monthly. When a database update is available you will receive an email notification that includes a link to download the updated database. You should install it as soon as possible so that you are using the most accurate data available.

A limited number of processes use Spectrum databases. Processes that use Spectrum databases include Spectrum Enterprise Tax, Spectrum Geocoding, Global Sentry, Spectrum Universal Addressing, and routing (from Spectrum Spatial and Routing). To view the processes you have installed that use Spectrum databases, open Spectrum Management Console and go to **Resources** > **Spectrum Databases** then click the Add button . If you have processes that use Spectrum databases, they are listed in the **Module** field.

Installing a Spectrum Database

Spectrum databases contain reference data from trusted data providers that is used to enhance and validate your data. For example, to perform address validation, Spectrum Technology Platform uses official address data from postal authorities to compare your address to addresses of record. Other types of processing that use Spectrum databases include geocoding, routing, and tax jurisdiction assignment for a given address.

We update Spectrum databases periodically to provide you with the most up-to-date data from third-party data providers. Database updates occur independently from software updates, in some cases quarterly or even monthly. When a database update is available you will receive an email notification that includes a link to download the updated database. You should install it as soon as possible so that you are using the most accurate data available.

Spectrum Technology Platform provides CLI commands that help you install, maintain, and archive SPD files.

- Use the **productdata archive register** on page 397 command to configure an alternate (non-default) location for the Spectrum Platform Data (SPD) archive.

- Use the **productdata extract register** on page 396 command to specify an alternate (non-default) extract location for a set of product data on the server.

  The default SPD file archival area is `../archive/ref-data`. Extracted SPD files are located in `../ref-data`.

- Use the **productdata install** on page 397 command to install your data files.

We recommend using the `productdata extract register` and `productdata archive register` commands to ensure that the SPD data, which may be large, resides in an appropriate and manageable location of your choosing on the server. The default location inside the Spectrum folders can make upgrades and re-installations more difficult and require even greater temporary space.

Adding a Spectrum Database

A Spectrum database contains reference data such as address data used to validate an address, or spatial data used for geocoding. To add a Spectrum database resource, see the instructions for the specific Spectrum product you are working with.

Database Pool Size and Runtime Instances

In most Spectrum Technology Platform environments there are multiple flows running at the same time, whether they are batch jobs or services responding to web service or API requests. To optimize concurrent processing, you can use the database pool size setting, which limits the number of concurrent requests a Spectrum database handles, and runtime instances, which controls the number of instances of a flow stage that run concurrently. These two settings should be tuned together to achieve optimal performance.

Database Pool Size

Spectrum databases contain reference data used by certain stages, such as postal data used to validate addresses, or geocoding data used to geocode addresses. These databases can be configured to accept multiple concurrent requests from the stages or services that use them, thereby improving the performance of the flows or service requests. The database pool size sets the maximum number of concurrent requests that a Spectrum database will process. By default, Spectrum databases have a pool size of 4, meaning the database can process four requests simultaneously.

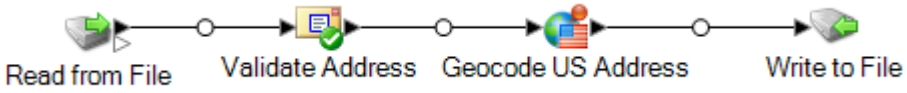
The optimal pool size varies by module. You will generally see the best results by setting the pool size between one-half to twice the number of CPUs on the server, with the optimal pool size for most

modules being the same as the number of CPUs. For example, if your server has four CPUs you may want to experiment with a pool size between 2 (one-half the number of CPUs) and 8 (twice the number of CPUs) with the optimal size possibly being 4 (the number of CPUs).

When modifying the pool size you must also consider the number of runtime instances specified in the flow for the stages accessing the database. Consider for example a flow that has a Geocode US Address stage that is configured to use one runtime instance. If you set the pool size for the US geocoding database to four, you will not see a performance improvement because there would be only one runtime instance and therefore there would only be one request at a time to the database. However, if you were to increase the number of runtime instances of Geocode US Address to four, you might then see an improvement in performance since there would be four instances of Geocode US Address accessing the database simultaneously, therefore using the full pool.

Runtime Instances

Each stage in a flow operates asynchronously in its own thread and is independent of any other stage. This provides for parallel processing of stages in a flow, allowing you to utilize more than one runtime instance for a stage. This is useful in flows where some stages process data faster than others. This can lead to an unbalanced distribution of work among the threads. For example, consider a flow consisting of these stages:



Depending on the configuration of the stages, it may be that the Validate Address stage processes records faster than the Geocode US Address stage. If this is the case, at some point during the execution of the flow all the records will have been processed by Validate Address, but Geocode US Address will still have records to process. In order to improve performance of this flow, it is necessary to improve the performance of the slowest stage - in this case Geocode US Address. One way to do that is to specify multiple runtime instances of the stage. Setting the number of runtime instances to two, for example, means that there will be two instances of that stage, each running in its own thread, available to process records. Keep in mind that while specifying multiple runtime instances can help improve performance, setting this value too high can strain your system resources, resulting in decreased performance.

Tuning Procedure

Finding the right settings for database pool size and runtime instances is a matter of experimenting with different settings to find the ones maximize available server resources without overloading resources and causing reduced performance.

> **Note:** You should optimize the flow pool size before tuning the database pool size. For information about optimizing the flow pool size, see **SettingDataflowPoolSize.dita#task_utx_h3t_tp**.

1. Begin by finding sample data to use as you test different settings. The sample dataset should be large enough that execution time is measurable and can be validated for consistency. The sample data should also be representative of the actual data you want to process. For example, if you are doing performance testing for geocoding, be sure that your test data has an equal number of records for all the countries you intend to geocode.
2. If you are testing a service or flow that requires the use of a database resource, such as postal databases or geocoding databases, make sure that you have the latest version of the database installed.
3. With sample data ready and the latest database resources installed, create a simple flow that reads data from a file, processes it with the stage you want to optimize, and writes to a file. For example, if you want to test performance settings for Validate Address, create a flow consisting of Read from File, Validate Address, and Write to File.
4. Set the database resource pool size to 1:
   a. Open Spectrum Management Console.
   b. Go to **Resources** > **Spectrum Databases**.
   c. Select the database resource you want to optimize and click the Modify button .
   d. In the **Pool size** field, specify 1.
   e. Click **OK**.
5. Set the stage's runtime instances to 1:
   a. Open the flow in Spectrum Enterprise Designer.
   b. Double-click the stage that you want to set to use multiple runtime instances.
   c. Click **Runtime**.

      Note: Not all stages are capable of using multiple runtime instances. If there is no **Runtime** button at the bottom of the stage's window, the stage is not capable of using multiple runtime instances.

   d. Select **Local** and specify 1.
   e. Click **OK** to close the **Runtime Performance** window, then click **OK** to close the stage.
6. Calculate baseline performance by running the flow several times and recording the average values for:
   - Elapsed time
   - CPU utilization
   - Memory utilization

   Tip: You can use the Spectrum JMX console to monitor performance. For more information, see **Monitoring Performance with the Spectrum JMX Console** on page 231.
7. Run multiple instances of the job concurrently, if this is a use case that must be supported. Record elapsed time, CPU utilization, and memory utilization for each scenario.

   Tip: You can use a file monitor to run multiple instances of a job at once. For more information, see **Triggering a Flow with a Control File** on page 189.

8. Increment the database resource pool size and the stage runtime instances setting.
9. Restart the server.
10. Run the flow again, recording the elapsed time, CPU utilization, and memory utilization.
11. Continue to increment the database resource pool size and the stage runtime instances until you begin to see diminishing performance.
12. If you are testing geocoding performance, repeat this procedure using single country and multicountry input.

Configuring database properties

Spectrum Technology Platform can access data that resides in remote/connected databases, bringing the data into memory for processing. The Spectrum Management Console **Add Database** window provides a UI-based method for configuring databases

Related concepts

Database Pool Size and Runtime Instances on page 150

Related reference

Running a Process Flow from the Command Line on page 206

Configure database connections

1. Access Spectrum Management Console using the URL `http://`*`server`*`:`*`port`*`/managementconsole`, where *server* is the server name or IP address, and *port* is the HTTP or HTTPS port used by Spectrum Technology Platform.

   The default port is 8080.
2. Go to **Resources** > **Spectrum Databases**.
3. Go to **Resources** > **Spectrum Databases**.
4. Click the **Add database** button to display the **Add Database** page.
5. In the **Name** field, give your database an easily recognizable name.
6. Define the database **Pool Size** and **Min/Max memory** settings.
   - **Max Memory** must be greater than zero, but cannot exceed 65336 MB/64GB.
   - **Min Memory** must be less than or equal to the **Max Memory** setting.

     Note: The fields for defining **Pool Size** and **Min/Max memory** values can be empty. If a value is empty, that value will not be specified on the command line when starting the component, as if no value were explicitly defined.

Note: For more information, see **Database Pool Size and Runtime Instances**.

7. Under **Module**, select the Spectrum Technology Platform module to which this database applies. This will display the available database types in the **Type** selection list.
8. Select the database to configure to populate the **Required Databases** and **Optional Databases** drop-down lists.

   Note that Optional Databases may not appear in the drop-down selection lists if none are available.
9. If necessary, select the **Override advanced settings** check box to adjust Java Properties, Environment Variables, or supply command line processing arguments.

Override advanced settings

This feature is available from the **Spectrum Databases > Add Database** window. It provides a way to define extra command line arguments in Spectrum Management Console, rather than setting up a property file that you would reference for processing. This makes it easier to define and maintain database settings across Spectrum Technology Platform, especially in a clustered environment.

Note: We suggest that you change these settings with caution, as they can affect processing globally.

- To add a new Java property or environment variable.
  a) Select either **Java Properties** or **Environment Variables** to expand the list of existing properties.
  b) Select the **Add** button +to display a blank input line.
  c) Add a **Name** and **Value** for the new entity.
  d) Click **Enter** to save the new entity.
- To delete a property or variable
  a) Select either **Java Properties** or **Environment Variables** to expand the list of existing definitions.
  b) Select the check box next to the entity to delete.
  c) Click the **Delete** button , above the entity list, to delete the setting.

  Note: You will not see a confirm message when you select the delete button.

- Specifying command line properties

  Use this field to define command line properties that are not memory-related settings and cannot be expressed as Java properties.

  Note: We suggest that you change these settings with caution, as they can affect processing globally.

Deleting a Spectrum Database

A Spectrum database contains reference data such as address data used to validate an address, or spatial data used for geocoding. You can delete Spectrum databases that are no longer used on your system. For example, you may want to delete a Spectrum database after installing an updated version of the database.

Important: Before deleting any resource, verify that there are no jobs or services using it. Deleting a resource that is referenced by jobs or services will cause those jobs or services to fail.

1. Open Spectrum Management Console.
2. Go to **Resources** > **Spectrum Databases**.
3. Put a check mark next to the Spectrum database you want to delete then click the Delete button .

Deleting a Spectrum database resource does not delete the database files themselves. After deleting the resource you must delete the database files if you want to free up space on your system.

6 - Services

In this section

Spectrum Services.....157
External Web Services.....160

Spectrum Services

A service is a processing capability that you access through the REST or SOAP web service interface or through the Spectrum Technology Platform API. You pass one or more records to the service and optionally specify the options to use when processing the record. The service processes the data and returns the data.

Some services become available when you install a Spectrum process. For example, when you install Spectrum Universal Addressing the service ValidateAddress becomes available on your system. In other cases, you must create a service in Spectrum Enterprise Designer then expose that service on your system as a user-defined service. For example, Spectrum Spatial services are unavailable until you create a service using a Spectrum Spatial stage.

Specifying Default Service Options

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

> **Note:** For a service, you can only modify default values before exposing the service for the first time. Once you expose the service you can no longer modify default values using Spectrum Enterprise Designer. Instead, you must use Spectrum Management Console.

1. Open Spectrum Management Console.
2. Click **Services**.
3. Check the box next to the service you want then click the Edit button .
4. Set the options for the service. For information about the service's options, see the solution guide for the service's module.
5. Click **Save**.

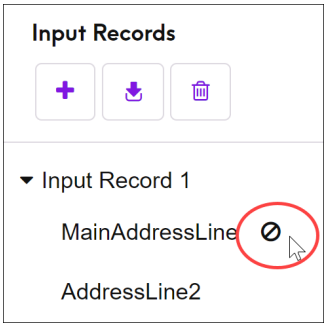
Previewing a Service

You can preview the results of a service in Spectrum 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.

1. Open Spectrum Management Console.
2. Go to the **Resources** 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:

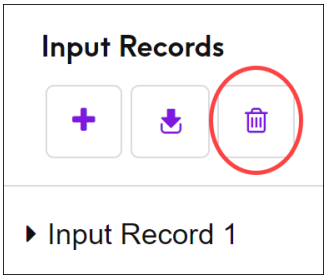
   - You do not have to enter data in every field. Leaving a field empty results in an empty string being used for preview.
   - If you want to preview the effect of passing a null value in a field, click the Disable icon next to the field:

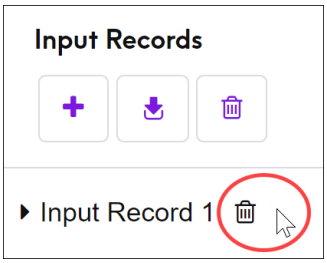
   - You can preview multiple records at once. To add a record, click the Add button .
   - You can import test data from a file. To import data, click the Import button . Note the following:
     - The first row in the file must be a header record. The field names in the header must match the field names required by the service.
     - 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:

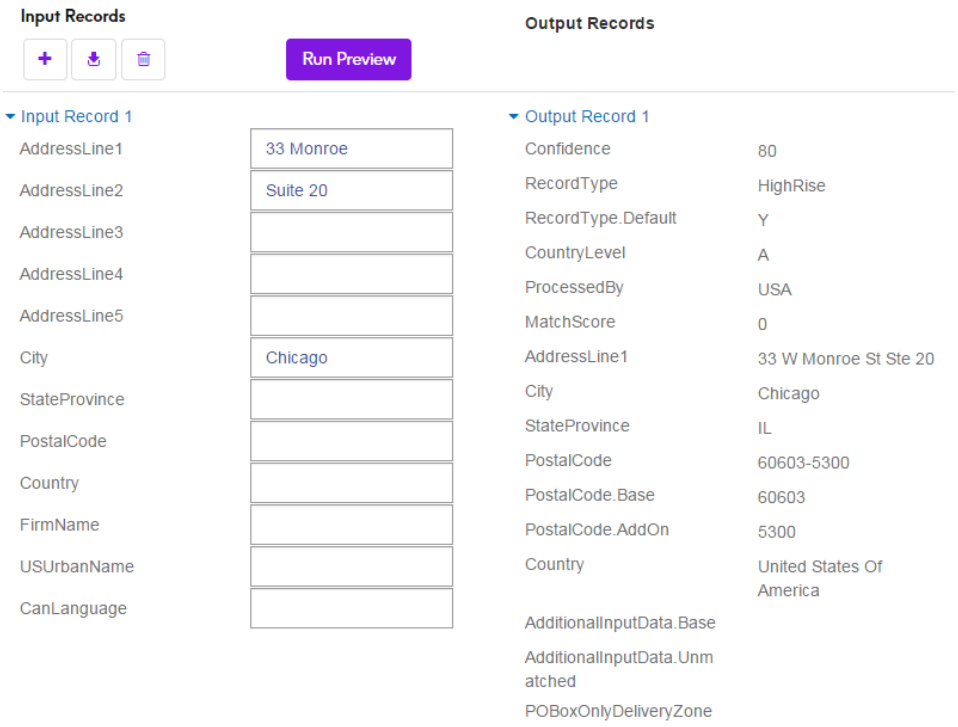
   - To delete an individual record, hover over the input record name (for example, "Input Record 1") and click the Delete button next to the record name:



- 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. If necessary you can make changes to the service's settings and click **Run Preview** again. (You do not need to input the data again.)

Optimizing Services

There are a number of different ways to call a Spectrum Technology Platform service and some of them perform better than others. The different ways of calling Spectrum Technology Platform services, in approximate order from fastest to slowest are:

- Client API over SOCKET
- Client API over HTTP
- Client API over HTTPS
- XML over HTTP
- Web Services - SOAP and REST over HTTP

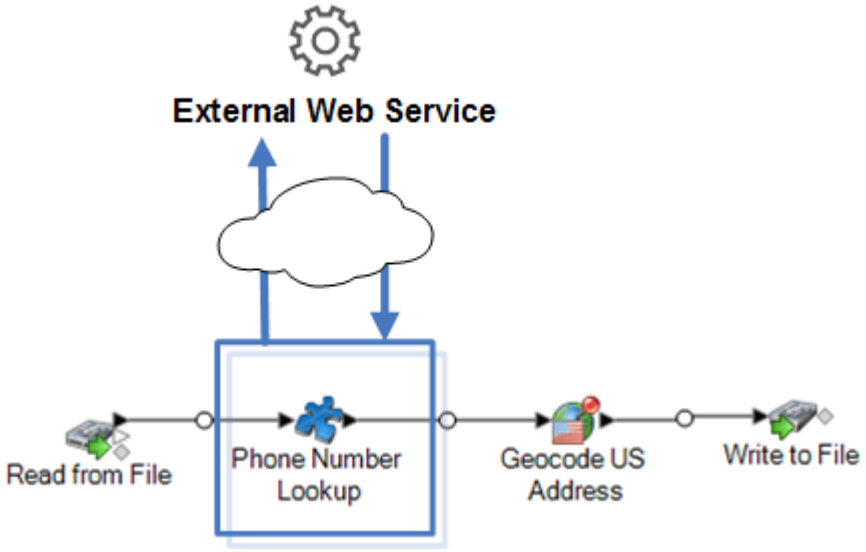
Invoking a service through the client API is generally faster than calling the web service. The network protocol can have a significant effect on the round-trip time for the service call. For example, using the persistent SOCKET connection instead of HTTP can improve response time by 30%-50%.

Performance in a real-time application calling Spectrum Technology Platform services also depends on whether the application is single-threaded or multithreaded, and whether the server has the resources available to fulfill the service request. For a single-threaded client application, specifying additional runtime instances of a stage will have minimal impact on response time. A multithreaded client application will generally benefit from multiple runtime instances, up to the number of concurrent threads.

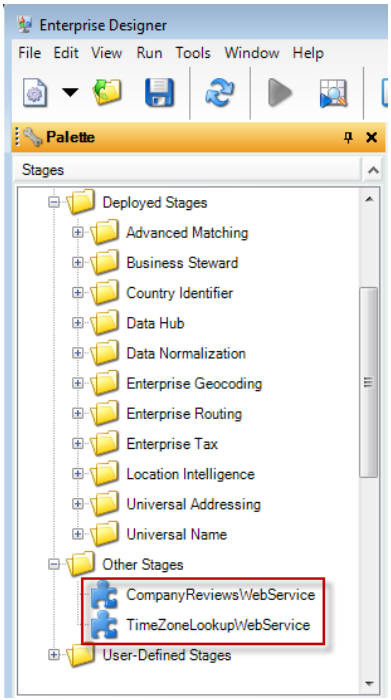
External Web Services

External web services are data processing services provided over the Internet by a third party. You can define external web services in Spectrum Management Console and use them as a stage in a dataflow. This allows you to incorporate almost any kind of processing into your Spectrum Technology Platform environment, since there are a wide variety of web services available on the Internet.

The following diagram illustrates the concept of external web services. Here, an external web service named Phone Number Lookup has been added to the dataflow. When the dataflow runs, Spectrum Technology Platform sends each record to the external web service. The external web service processes the record and returns to the stage. The updated record, with the phone number added, continues to the next stage in the dataflow, in this example Geocode US Address.



External web services show up in the palette in Spectrum Enterprise Designer and you can work with them as you do other stages. The following shows an example of two external web services, CompanyReviewsWebService and TimeZoneLookupWebService.



Requirements and Limitations

Spectrum Technology Platform supports external web services that use REST, SOAP 1.1, or SOAP 1.2 messaging, with the following limitations:

- WADL requests and responses with more than one representation are not supported.

- Recursive schemas are not supported.

Adding an External Web Service

External web services are data processing services provided over the Internet by a third party. You can use an external web service as a stage in a flow, allowing you to expand the capabilities of your Spectrum Technology Platform server.

This procedure defines a connection between your Spectrum Technology Platform server and a third-party web service. After completing this procedure, you will have a new stage in Spectrum Enterprise Designer which represents the external web service. You can then use the external web service as you would any other stage in a flow.

1. Open Spectrum Management Console.
2. Go to **Resources** > **External Web Services**.
3. Click the Add button .
4. On the **Descriptor** step:
   a) Specify the web service's WSDL or WADL by one of these methods:

To load the web service's descriptor from a URL	In the **Load descriptor** field, choose **By URL** and in the **URL** field specify the URL of the WADL or WSDL. Enter your credentials for the web service if required by the web service.
To load the web service's descriptor from a file	In the **Load descriptor** field, choose **Upload** then select the file in the **Upload file** field. Some web service venders prefer to provide the WSDL or WADL as a file rather than making it available via a URL.
For REST web services that have no WADL	In the **Load descriptor** field, choose **None**.

   b) Click **Next**.
5. On the **Settings** step:
   a) In the **Name** field, enter the name you want to give the external web service when it is exposed on Spectrum Technology Platform. This will be the stage name shown in Spectrum Enterprise Designer. The name can be anything you want but must not already be used by another web service on your system.
   b) In the **Timeout** field, enter the number of seconds that are allowed to elapse before a request submitted to the web service times out.

      Note: The timeout value you specify here applies to any request made to the web service. This includes not only transactions made against the exposed web service,

but also requests made while configuring the web service. Such configuration requests are invoked by choosing a new item on the **Request** page and running preview. Timeouts can occur when performing any of these actions. If timeouts do occur, increasing the **Timeout** value can help to avoid them, provided the web service is actually up and running.

c) If the external web service requires a user name and password, under **Security settings**, in the **Type** field, choose how to transmit the user name and password from the Spectrum Technology Platform server to the external web service.

None	Choose this option if you do not need to provide a user name and password to use the external web service.
Basic Authentication	Choose this option to pass user name and password information to the external web service in the HTTP header.
WS-Security	For SOAP services only, choose this option to pass user name and password information to the external web service through the header of the SOAP message.

d) Enter a user name and password if they are required to access the external web service.

e) Click **Next**.

6. On the **Request** step, configure the parameters to include in requests to the external web service.

For REST web services:

URL If you did not supply a WADL on the **Descriptor** step, enter a sample request URL containing the path parameters (if any) and query parameters that you want to include in the request. For example:

http://example.com/rest/customers/{state}?age=31

Here, there is one path parameter, `{state}`, and one query parameter, `age`.

If you supplied a WADL on the **Descriptor** step, the **URL** field displays the endpoint based on the WADL you provided. You cannot edit the endpoint.

Resource This setting is only visible if you supplied a WADL on the **Descriptor** step. Select the web service resource that you want to expose on Spectrum Technology Platform.

> **Note:** If you want to expose more than one resource, you must define a separate external web service for each resource.

Method Select the HTTP method to use to for the request to the external web service.

If you provided a WADL on the **Descriptor** step, only those HTTP methods supported by the external web service are listed.

Path parameters

The **Path parameters** section lists the parameters that are part of the URL path, if the external web service uses path parameters. For example, this URL contains the path parameter `{state}`:

http://example.com/rest/customers/{state}?age=31

If you provided a WADL on the **Descriptor** step, the web service's path parameters are listed. If you did not supply a WADL on the **Descriptor** step, the list of path parameters is generated from the sample request URL you provided in the **URL** field. To add or remove a path parameter, add or remove it from the URL. Anything in the URL surrounded by curly braces is interpreted as a path parameter.

Query parameters

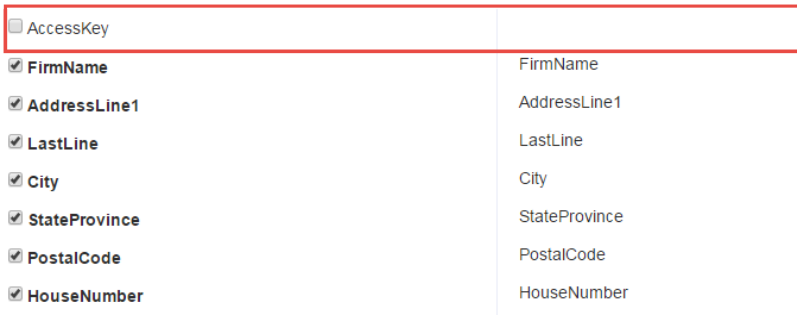
The **Query parameters** section lists the parameters that appear after the "?" in the request URL. For example, this URL contains the query parameter `age`:

http://example.com/rest/customers/{state}?age=31

If you provided a WADL on the **Descriptor** step, the web service's query parameters are listed. If you did not provide a WADL on the **Descriptor** step, the list of query parameters is generated from the sample request URL you provided in the **URL** field. To add or remove a query parameter, add or remove it from the URL.

Table 2: Settings for REST Path Parameters and Query Parameters

Setting	Description
Expose	Check the box in this column to make the parameter available in the Spectrum Technology Platform stage.
Request	This column lists the parameter name that is used in the request to the external web service.
Input	This column lists the input field names as they will appear in a flow. You can leave the names the same as those used by the third-party web service or you can change them by hovering over the name and clicking the edit button.

Setting	Description
Default Value	Check the box in this column if you want to specify a default value for the field. Enter the default value in to the field that appears after checking the box. If you want to provide a default value that cannot be overridden from a flow, check the box in the **Default Value** column and clear the corresponding box in the **Request** column. This hides the field from flows while providing the default value in the request to the external web service. This may be useful if you have an access key that you need to provide in each request. For example: 

For SOAP web services:

URL This field displays the endpoint based on the WSDL you provided on the **Descriptor** step. You cannot edit this field.

Operation Select the web service operation you want to perform.

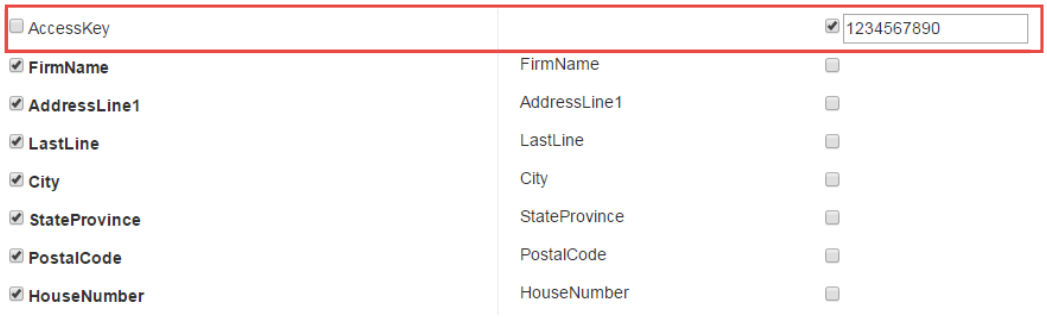
> **Note:** If you want to expose more than one operation, you must define a separate external web service for each operation.

Request In this column, select the fields and options you want to make available through Spectrum Technology Platform.

Input This column lists the input field names as they will appear in a flow. You can leave the names the same as those used by the third-party web service or you can change them by hovering over the name and clicking the edit button .

Default Value Check the box in this column if you want to specify a default value for the field. Enter the default value in to the field that appears after checking the box.

If you want to provide a default value that cannot be overridden from a flow, check the box in the **Default Value** column and clear the corresponding box in the **Request** column. This hides the field from flows while providing the default value in the request to the external web service. This may be useful if you have an access key that you need to provide in each request. For example:



7. If you selected POST or PUT in the **Method** field, define the structure of the data you want to send to the web service in the POST or PUT operation. To do this, click the **Format** button and choose one of the following options:

   Upload schema — Choose this option if you have an XML schema that defines the structure of the data you to send to the web service in the POST or PUT operation. After selecting this option, browse to the schema file.

   Provide sample — Choose this option if you have a sample of the data you want to send to the web service in the POST or PUT operation. After selecting this option, you can enter the sample manually or paste the sample into the window.

   After you provide a schema or sample, check the box next to each data element you want to make available in the flow stage.

8. Click **Next**.
9. On the **Headers** step:
   a) Under **HTTP headers**, specify the value to pass to the external web service for each header. The values you specify here are used for all requests from Spectrum Technology Platform to the external web service. If no headers are listed, the external web service does not require any HTTP headers.
   b) For SOAP web services, if the external web services supports SOAP headers you can select the headers you want to use under **SOAP headers**. You can specify a default value for each SOAP header. The default value can be overridden in each request to Spectrum Technology Platform. The **Input** column shows the name of the header that will be used in requests to Spectrum Technology Platform. If you want to use a different name, hover over the name and click the edit button.

      If a check box is checked and grayed out, it means that the header is required and you cannot disable it.

   c) Click **Next**.
10. On the **Response** step:
    a) If you want the response from the external web service to be returned in a single field, check the box **Return payload as field**. All response elements will be placed in a single field instead of being returned in separate fields. For REST web services, the field name is RestReponse, and for SOAP web services the field name is SoapResponse.

b) If you are configuring a REST web service, there is a **Format** button. Click this button to choose how you want to define the structure of the web service response that will be returned by Spectrum Technology Platform:

Upload schema	Choose this option if you have an XML schema that defines the structure of the response you want Spectrum Technology Platform to return. After selecting this option, browse to the schema file.
Provide sample	Choose this option if you have a sample response from the external web service and you want to use it to define the response you want from Spectrum Technology Platform. After selecting this option, you can enter the sample manually or paste the sample into the window.

c) In the **Response** column, choose the fields you want to make available in Spectrum Technology Platform.

The icon indicates that a field will be returned as a *list* field. A list is a Spectrum Technology Platform data type that contains hierarchical data that can be repeated. For example, a field PhoneNumbers may contain multiple Phone fields:

```
<PhoneNumbers>
   <Phone>
 <Type>Cell</Type>
 <Number>312-123-4567</Number>
   </Phone>
   <Phone>
 <Type>Home</Type>
 <Number>773-123-4567</Number>
   </Phone>
</PhoneNumbers>
```

In this case the PhoneNumbers field would be a list field that can contain a list of Phone elements.

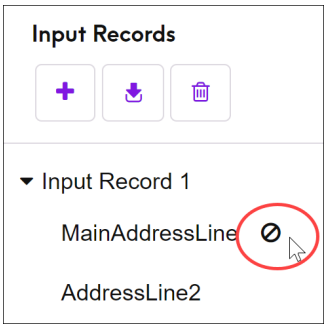
d) The **Output** column lists the output field names as they will appear in a flow. You can leave the names the same as those used by the third-party web service or you can change them by hovering over the name and clicking the edit button .

e) Click **Next**.

11. On the **Preview** step, you can test the external web service by entering sample data then clicking **Run Preview**. This is optional.

Here are some tips for entering sample data:

- 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 the default value or a null value in a field, click the Disable icon next to the field:

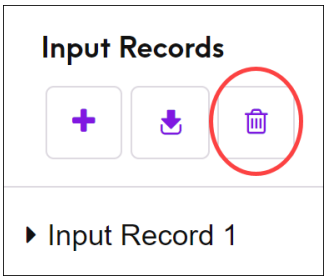


If you have defined a default value for the field on the **Request** tab, the default value will be used. If you have not defined a default value, a null value will be used.

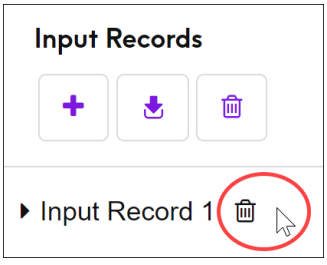
- You can preview multiple records at once. To add a record, click the Add button .
- You can import test data from a file. To import data, click the Import button . Note the following:
  - The first row in the file must be a header record. The field names in the header must match the field names required 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:



12. To make the external web service available to use in flows, switch the **Enabled** switch to **On**.

13. Click **Save**.

The external web service is now defined and available to use as a stage in a flow in Spectrum Enterprise Designer.

Previewing an External Web Service

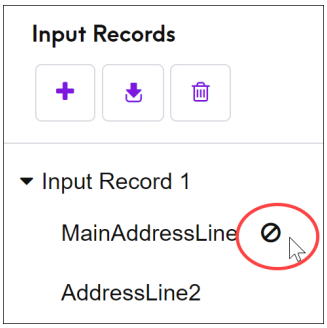
You can preview the response from an external web service by sending a test request. This can be done while adding a new external web service, in the last step of the "Add Web Service" wizard. You can also preview the response from external web services that have already been added to your Spectrum Technology Platform server. This topic describes how to preview an external web service that has already been added.

> **Note:** In order to be able to preview an external web service, you must have View and Execute privileges for **Platform - Services** in addition to View and Modify privileges for **External Web Services - Connection**.

1. In Spectrum Management Console, go to **Services** > **External Web Services**.
2. Click the external web service you want to preview.
3. Enter the sample data you want to use in the test request to the external web service.

   Here are some tips for entering sample data:

   - 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 the default value or a null value in a field, click the Disable icon next to the field:

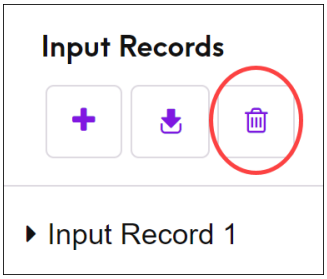
     

     If you have defined a default value for the field on the **Request** tab, the default value will be used. If you have not defined a default value, a null value will be used.
   - You can preview multiple records at once. To add a record, click the Add button .
   - You can import test data from a file. To import data, click the Import button . 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.

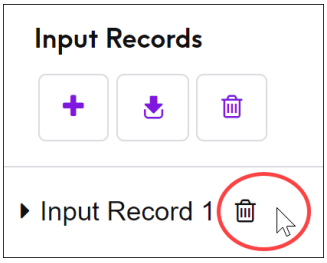
- If the file uses a space as the field separator, field values must be surrounded by quotes. Here is an example of a file that uses a space as the field separator:

```
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:



4. Click **Run Preview**.

The results from the external web service are displayed to the right of the input data.

Exporting an External Web Service Definition

An external web service definition contains the connection properties that enable Spectrum Technology Platform access a third-party web service over the Internet. Connection properties include information like the external web service's URL and your credentials. You can save this information to a file so that you can easily add the external web service to another Spectrum Technology Platform server.

1. Open Spectrum Management Console.
2. Go to **Resources** > **External Web Services**.
3. Check the box next to the external web service definition that you want to export then click the Export button .
4. Choose a location to save the file.

The external web service definition is saved as a file with a `.ews` file extension.

Importing an External Web Service Definition

An external web service definition contains the connection properties that enable Spectrum Technology Platform access a third-party web service over the Internet. Connection properties include information like the external web service's URL and your credentials. You can import an external web services definition file, which enables you to take an external web services definition from one Spectrum Technology Platform server and implement it on another server.

1. Log in to Spectrum Management Console on the server onto which you want to import the web service definition.
2. Go to **Resources** > **External Web Services**.
3. Click the Import button .
4. Select the external web services definition file you want to import. The external web services definition file has an `.ews` file extension.
5. Click **OK**

Deleting an External Web Service

Deleting an external web service will break existing flows that reference the external web service. If no flows reference the external web service you can safely delete the external web service.

1. In Spectrum Management Console, go to **Resources** > **External Web Services**.
2. Check the box next to the web service you want to delete then click the Delete button .
3. Click **OK** to confirm.

7 - Flows

In this section

Configuring Flow Defaults....173
Scheduling Flows....182
Viewing Flow Status and History....185
Triggering a Flow with a Control File....189
Command Line Execution....193
Adding Flow Runtime Options....210

Configuring Flow Defaults

Setting Data Type Conversion Defaults

You can set the default data type conversion settings for your system in Spectrum Management Console. You can override the default formats for individual dataflows in Spectrum Enterprise Designer.

To set the default data type conversion options for your system, follow this procedure.

1. Open Spectrum Management Console.
2. Go to **Flows** > **Defaults**.
3. Click **Data Type Conversions**.
4. Specify the formats that you want to use for date and time data that is converted to a string. When the data or time is converted to a string, the string will be in the format you specify here.
   a) In the **Locale** field, select the country whose format you want to use for dates converted to a string. Your selection will determine the default values in the **Date**, **Time**, and **DateTime** fields. Your selection will also determine the language used when a month is spelled out. For example, if you specify English the first month of the year would be "January" but if you specify French it would be "Janvier."
   b) In the **Date** field, select the format to use for date data when it is converted to a string. A list of the most commonly used formats for the selected locale is provided.

      For example, if you choose the format **M/D/YY** and a date field contains 2020-3-2, that date data would be converted to the string `3/2/20`.

   c) In the **Time** field, select the format to use for time data when it is converted to a string. A list of the most commonly used formats for the selected locale is provided.

      For example, if you choose the format **h:mm a** and a time field contains 23:00, that time data would be converted to the string `11:00 PM`.

   d) In the **DateTime** field, select the format to use for fields containing the DateTime data type when converted to a string. A list of the most commonly used formats for the selected locale is provided.

      For example, if you choose the format **M/d/yy h:mm a** and a DateTime field contains 2020-3-2 23:00, that DateTime data would be converted to the string `3/2/20 11:00 PM`.

   e) In the **Whole numbers** field, select the formatting you want to use for whole numbers (data types float and double).

For example, if you choose the format **#,###** then the number 4324 would be formatted as `4,324`.

Note: If you leave this field blank, numbers will be formatted in the same way they were in Spectrum Technology Platform 8.0 and earlier. Specifically, no thousands separator is used, the dot (".") is used as the decimal separator, numbers less than 10^{-3} or greater than or equal to 10^{7} are shown in scientific notation, and negative numbers have a minus sign ("-") in front of them. Also note that if you leave this field blank, numbers that use the bigdecimal data type will always be in the format #,###.000.

f) In the **Decimal numbers** field, select the formatting you want to use for numbers that contain a decimal value (data types integer and long).

For example, if you choose the format **#,##0.0#** then the number 4324.25 would be formatted as `4,324.25`.

Note: If you leave this field blank, numbers will be formatted in the same way they were in Spectrum Technology Platform 8.0 and earlier. Specifically, no thousands separator is used, the dot (".") is used as the decimal separator, numbers less than 10^{-3} or greater than or equal to 10^{7} are shown in scientific notation, and negative numbers have a minus sign ("-") in front of them. Also note that if you leave this field blank, numbers that use the bigdecimal data type will always be in the format #,###.000.

You can also specify your own date, time, and number formats if the ones available for selection do not meet your needs. To specify your own date or time format, type the format into the field using the notation described in **Date and time patterns** on page 175. To specify your own number format, type the format into the file using the notation described in **Number Patterns** on page 178.

5. Under **Null handling**, choose whether to perform type conversion if a field contains a null value. If you select any of the following options, either the dataflow or the record containing the null value will fail based on your selection in the **Failure handling** field.

Fail null string	Fail the flow or record if type conversion is needed on a string field that contains a null value.
Fail null Boolean	Fail the flow or record if type conversion is needed on a Boolean field that contains a null value.
Fail null numeric	Fail the flow or record if type conversion is needed on a numeric field that contains a null value. Numeric fields include double, float, long, integer, and Big Decimal fields.
Fail null date	Fail the flow or record if type conversion is needed on a date field that contains a null value. This includes date, time, and DateTime fields.

6. In the **Failure handling** field, specify what to do when a field's value cannot be automatically converted to the data type required by a stage.

Fail the flow	If a field cannot be converted the flow will fail.
Fail the record	If a field cannot be converted the record will fail but the flow will continue to run.
Initialize the field using default values	If a field cannot be converted the field's value is replaced with the value you specify here. This option is useful if you know that some records contain bad data and you want to replace the bad data with a default value. Specify a value for each data type.

Date and time patterns

When defining data type options for date and time data, you can create your own custom date or time pattern if the predefined ones do not meet your needs. To create a date or time pattern, use the notation described in the table below. For example, this pattern:

dd MMMM yyyy

Would produce a date like this:

14 December 2020

Letter	Description	Example
G	Era designator	AD
yy	Two-digit year	96
yyyy	Four-digit year	1996
M	Numeric month of the year.	7
MM	Numeric month of the year. If the number is less than 10 a zero is added to make it a two-digit number.	07
MMM	Short name of the month	Jul
MMMM	Long name of the month	July
w	Week of the year	27
ww	Two-digit week of the year. If the week is less than 10 an extra zero is added.	06

Letter	Description	Example
W	Week of the month	2
D	Day of the year	189
DDD	Three-digit day of the year. If the number contains less than three digits, zeros are added.	006
d	Day of the month	10
dd	Two-digit day of the month. Numbers less than 10 have a zero added.	09
F	Day of the week in month	2
E	Short name of the day of the week	Tue
EEEE	Long name of the day of the week	Tuesday
a	AM PM marker	PM
H	Hour of the day, with the first hour being 0 and the last hour being 23.	0
HH	Two-digit hour of the day, with the first hour being 0 and the last hour being 23. Numbers less than 10 have a zero added.	08
k	Hour of the day, with the first hour being 1 and the last hour being 24.	24
kk	Two-digit hour of the day, with the first hour being 1 and the last hour being 24. Numbers less than 10 have a zero added.	02
K	Hour hour of the morning (AM) or afternoon (PM), with 0 being the first hour and 11 being the last hour.	0
KK	Two-digit hour of the day, with the first hour being 1 and the last hour being 24. Numbers less than 10 have a zero added.	02
h	Hour of the morning (AM) or afternoon (PM), with 1 being the first hour and 12 being the last hour.	12

Letter	Description	Example
hh	Two-digit hour of the morning (AM) or afternoon (PM), with 1 being the first hour and 12 being the last hour. Numbers less than 10 have a zero added.	09
m	Minute of the hour	30
mm	Two-digit minutes of the hour. Numbers less than 10 have a zero added.	05
s	Second of the minute	55
ss	Two-digit second of the minute. Numbers less than 10 have a zero added.	02
S	Millisecond of the second	978
SSS	Three-digit millisecond of the second. Numbers containing fewer than three digits will have one or two zeros added to make them three digits.	978 078 008
z	Time abbreviation of the time zone name. If the time zone does not have a name, the GMT offset.	PST GMT-08:00
zzzz	The full time zone name. If the time zone does not have a name, the GMT offset.	Pacific Standard Time GMT-08:00
Z	The RFC 822 time zone.	-0800
X	The ISO 8601 time zone.	-08Z
XX	The ISO 8601 time zone with minutes.	-0800Z
XXX	The ISO 8601 time zone with minutes and a colon separator between hours and minutes.	-08:00Z

Number Patterns

When defining data type options for numeric data, you can create your own custom number pattern if the predefined ones do not meet your needs. A basic number pattern consists of the elements below:

- A prefix such as a currency symbol (optional)
- A pattern of numbers containing an optional grouping character (for example a comma as a thousands separator)
- A suffix (optional)

For example, this pattern:

$ ###,###.00

Would produce a number formatted like this (note the use of a thousands separator after the first three digits):

$232,998.60

Patterns for Negative Numbers

By default, negative numbers are formatted the same as positive numbers but have the negative sign added as a prefix. The character used for the number sign is based on the locale. The negative sign is "-" in most locales. For example, if you specify this number pattern:

```
0.00
```

The number negative ten would be formatted like this in most locales:

```
-10.00
```

However, if you want to define a different prefix or suffix to use for negative numbers, specify a second pattern, separating it from the first pattern with a semicolon (";"). For example:

```
0.00;(0.00)
```

In this pattern, negative numbers would be contained in parentheses:

```
(10.00)
```

Scientific Notation

If you want to format a number into scientific notation, use the character `E` followed by the minimum number of digits you want to include in the exponent. For example, given this pattern:

```
0.###E0
```

The number `1234` would be formatted like this:

```
1.234E3
```

In other words, 1.234 x 10^3.

Note that:

- The number of digit characters after the exponent character gives the minimum exponent digit count. There is no maximum.
- Negative exponents are formatted using the localized minus sign, not the prefix and suffix from the pattern.
- Scientific notation patterns cannot contain grouping separators (for example, a thousands separator).

Special Number Pattern Characters

The characters below render other characters, as opposed to being reproduced literally in the resulting number. If you want to use any of these special charters as literal characters in your number pattern's prefix or suffix, surround the special character with quotes.

Symbol	Description

`0`	Represents a digit in the pattern including zeros where needed to fill in the pattern. For example, the number twenty-seven when applied to this pattern:
	`0000`
	Would be:
	`0027`

`#`	Represents a digit but zeros are omitted. For example, the number twenty-seven when applied to this pattern:
	`####`
	Would be:
	`27`

`.`	The decimal separator or monetary decimal separator used in the selected locale. For example, in the U.S. the dot (.) is used as the decimal separator but in France the comma (,) is used as the decimal separator.
`-`	The negative sign used in the selected locale. For most locals this is the minus sign (-).

Symbol	Description

,	The grouping character used in the selected locale. The appropriate character for the selected locale will be used. For example, in the U.S., the comma (,) is used as a separator.
	The grouping separator is commonly used for thousands, but in some countries it separates ten-thousands. The grouping size is a constant number of digits between the grouping characters, such as 3 for 100,000,000 or 4 for 1,0000,0000. If you supply a pattern with multiple grouping characters, the interval between the last one and the end of the integer is the one that is used. For example, all the following patterns produce the same result:
	`#,##,###,####`
	`######,####`
	`##,####,####`

E	Separates mantissa and exponent in scientific notation. You do not need to surround the E with quotes in your pattern. See **Scientific Notation** on page 178.
;	Separates positive and negative subpatterns. See **Patterns for Negative Numbers** on page 178.

%	Multiply the number by 100 and show the number as a percentage. For example, the number .35 when applied to this pattern:
	`##%`
	Would produce this result:
	`35%`

¤	The currency symbol for the selected locale. If doubled, the international currency symbol is used. If present in a pattern, the monetary decimal separator is used instead of the decimal separator.

'	Used to quote special characters in a prefix or suffix. For example,
	`"'#'#"`
	Formats 123 to:
	`"#123"`
	To create a single quote itself, use two in a row:
	`"# o''clock"`

Setting the Malformed Records Default

A malformed record is an input record that Spectrum Technology Platform cannot parse. By default, if the input data for a job contains one malformed record, the job will stop. You can change this setting to allow more malformed input records, or even allow an unlimited number of malformed records. This procedure describes how to set a default malformed record threshold for jobs on your system.

> **Note:** You can override the default malformed record threshold for a job by opening the job in Spectrum Enterprise Designer and going to **Edit** > **Job Options**.

1. Open Spectrum Management Console.
2. Go to **Flows** > **Defaults**.
3. Click **Malformed Records**.
4. Select one option:

Stop jobs containing this many malformed records	Select this option to stop jobs if the input data contains one or more malformed records. Enter the number of malformed records that you want to trigger the termination of the job. The default is 1.
Do not stop flows with malformed records	Select this option to allow an unlimited number of malformed records in the input data.

Setting Report Defaults

Reports are generated by jobs that contain a report stage. Reports can include processing summaries such as the number of records processed by the job, or postal forms such as the USPS CASS 3553 form. Some modules come with predefined reports. You can also create custom reports. Setting report defaults establishes the default settings for saving reports. The default settings can be overridden for a job, or for a particular report with in a job, by using Spectrum Enterprise Designer.

This procedure describes how to set the default reporting options for your system.

1. Open Spectrum Management Console.
2. Go to **Flows** > **Defaults**.
3. Click **Reports**.
4. Choose the format you want to use to save reports. Reports can be saved as HTML, PDF, or text.
5. Choose where you want to save reports.

Save reports to job history	Saves reports on the server as part of the job history. This makes it convenient for Spectrum Management Console and Spectrum Enterprise

Designer users to view reports since the reports are available in the execution history.

Save reports to a file Saves reports to a file in the location you specify. This is useful if you want to share reports with people who are not Spectrum Technology Platform users. It is also useful if you want to create an archive of reports in a different location. To view reports saved in this manner you can use any tool that can open the report's format, such as a PDF viewer for PDF reports or a web browser for HTML reports.

6. If you selected **Save reports to a file**, complete these fields.

Report location The folder where you want to save reports.

Append to report name Specifies variable information to include in the file name. You can choose one or more of these options:

- **Job ID** A unique ID assigned to a job execution. The first time you run a job on your system the job has an ID of 1. The second time you run a job, either the same job or another job, it has a job ID of 2, and so on.
- **Stage** The name of the stage that contributed data to the report, as specified in the report stage in Enterprise Designer.
- **Date** The day, month, and year that the report was created.

Overwrite existing reports Replaces previous reports that have the same file name with the new report. If you do not select this option and there is an existing report that has the same name as the new report, the job will complete successfully but the new report will not be saved. A comment will appear in the execution history indicating that the report was not saved.

Scheduling Flows

Scheduling a Flow

Scheduling a flow enables a job or dataflow to run automatically at a specified time or times.

Note: Scheduling a recurring date and time: All flows start the first day of each month, and repeat according to the recurring schedule you set. Recurrence scheduling defines the start and end time and interval on which flows will run. For example, if you schedule a flow to run

every six days, at 2:00 AM, the flow will run day 1, day 6, day 12, day 24, through the end of the month, at the same time.

- On a specific date and time
- On a recurring basis on the specified dates and times

Note: In order to create, edit, or view a schedule, you must have View permission for the secured entity type being scheduled, either **Dataflows** or **Process Flows**.

1. If you have not already done so, expose the flow.

   You can expose a flow by opening it in Spectrum Enterprise Designer and selecting **File** > **Expose/Unexpose and Save**.

2. Open Spectrum Management Console.
3. Go to **Flows** > **Schedules**.
4. Click the Add button .
5. In the **Name** field, enter the name you want to give to this schedule. This is the name that will be displayed in the schedules listing.
6. In the **Flow** field, enter the job or process flow that you want to run. Only jobs and process flows that are saved and exposed are available here.
7. After you specify a flow, additional fields appear below the **Flow** field, one field for each of the flow's source stages (such as Read from File) and sink stages (such as Write to File).

   These fields show the files that will be used when the flow runs by this schedule. By default, the flow will use the files specified in the flow's sources and sinks. You can specify different files to use when this schedule runs by replacing the file path with the path to another file. For example, if your flow has a Read from File stage that reads data from `C:\FlowInput\Customers.csv` but you want to use data from `C:\FlowInput\UpdatedCustomers.csv` when this schedule runs, you would specify `C:\FlowInput\UpdatedCustomers.csv` in the Read from File field.

   Note: In order change the files used in the source and sink stages you must have Read permission for the **Resources - File Servers** secured entity type.

   Note that when a flow is triggered by a schedule the files used by a flow must reside on the Spectrum Technology Platform server or on a file server defined as an external resource in Spectrum Management Console. This applies both to jobs as well as job activities within a process flow. If a source or sink stage references a file on a client computer to modify the dataflow or override the dataflow file location.

   Option 1: Move the file to the Spectrum Technology Platform server or file server then modify the dataflow:

   a) Open the dataflow in Spectrum Enterprise Designer.
   b) Double-click the source or sink stage.
   c) In the **File name** field, click the browse button.
   d) Click **Remote Machine** then select the file you want.

Note: If you are running Spectrum Enterprise Designer on the same machine as the Spectrum Technology Platform server, it will appear that clicking Remote Machine is no different than clicking My Computer. However, you must select the file using Remote Machine in order for the system to recognize the file as being on the Spectrum Technology Platform server.

Option 2: Override the dataflow file location when this schedule runs.

You can override the file references contained in the flow when this schedule runs. To do this, replace the default file shown in each source and sink field with a path to a file on the Spectrum Technology Platform server or a file server resource defined in Spectrum Management Console.

8. In the **Trigger** field, select one of these options:

Date/Time	Run the flow once at a specific date and time.
Recurring Date/Time	Run the flow on multiple dates and times using a pattern of recurrence.
Control File	Run the flow when a file appears in a specified directory. For information about using a control file, see **Triggering a Flow with a Control File** on page 189.

9. Specify the date and time or the recurrence interval for running the flow.

   Note: If you chose **Recurring Date/Time** in the **Trigger** field, be sure to select a start date that conforms to the recurrence pattern. For example, if you chose to run the flow on the first Monday of the month, be sure to select a date that is the first Monday of the month. If you select a date that does not meet the recurrence pattern, the flow may run at an unexpected time. Also, selecting a start date in the past may result in the flow running at an unexpected time.

10. If the flow is configured to send email notifications you can specify additional recipients for the notifications that will be sent when this schedule runs. The recipients you specify here will receive notifications in addition to those recipients specified in the flow's notification settings. To configure a flow to send notifications, open the flow in Spectrum Enterprise Designer and go to **Edit** > **Notifications**.
11. Click **Save**.

Viewing Schedules

A flow schedule defines when a job or process flow will run. You can view the flow schedules on your system as well as the results of each run.

1. Open Spectrum Management Console.
2. Go to **Flows** > **Schedules**.

A list of the flow schedules is displayed. To sort the list of schedules, click the column heading. You can filter the list by typing keywords into the **Filter** field. Note that the filter field only filters on the **Schedule Name**, **User**, **Next Run**, and **Last Run** columns.

Deleting a Schedule

A scheduled flow runs automatically at a set time or on a repeating schedule. If you no longer want a flow to run on a schedule, you can delete the schedule. Deleting a schedule does not delete the flow itself.

To delete a schedule:

1. Open Spectrum Management Console.
2. Go to **Flows** > **Schedules**.
3. Check the box next to the schedule you want to delete then click the Delete button .

Viewing Flow Status and History

You can view a history of job, process flow, and service execution in Spectrum Management Console and Spectrum Enterprise Designer.

In Spectrum Management Console

To view flow status and history in Spectrum Management Console, go to **Flows** > **History**. The **Flows** tab shows job and process flow history, and the **Transactions** tab shows services history.

> **Note:** For flow history, the record counts shown when you hover over the **Results** column reflect the total number of records written as output by all the flow sinks. This number may differ from the number of input records if the flow combines records, splits records, or creates new records.

By default, transaction history is disabled because enabling transaction history can have an adverse impact on performance. If you want to see transaction history you must turn on transaction history logging by clicking the **Transaction logging** switch. To view user activity, consider using the audit log which you can access under **System** > **Logs**.

The flow history list updates automatically every 30 seconds. If you want to update it sooner, click the Refresh button .

In Spectrum Enterprise Designer

To view flow status and history in Spectrum Enterprise Designer, go to **View** > **Execution History**.

The flow history list updates automatically every 30 seconds. If you experience slowness when viewing execution history uncheck the **Auto refresh** box.

The **Jobs** tab is used to monitor job status and to pause, resume, or cancel jobs that are running as well as delete completed jobs.

> **Note:** The record counts shown on the **Jobs** tab reflect the total number of records written as output by all the flow sinks. This number may differ from the number of input records if the flow combines records, splits records, or creates new records.

- The **Succeeded** column shows the total number of records written as output by all the flow sinks that have an empty value in the Status field.
- The **Failed** column shows the total number of records written as output by the flow sinks that have a value of F in the Status field.
- The **Malformed** column shows the total number of records coming out of all source stage error ports.

The **Process Flows** tab is used to monitor process flow status and to cancel process flows that are running as well as delete completed process flows. If you click the plus sign next to any given process flow, you will view Activity Status information for the process flow. This information is included in this area:

ActivityName	Includes the names of all activities, including any success activities, that make up the process flow.	
State	The status of the activity (failed, succeeded, running, canceled).	
ReturnCode	A code that indicates the result of the process flow:	
	1	The process flow failed.
	0	The process flow finished successfully.
	-1	The process flow was canceled.
	Other numbers	If the process flow contains a Run Program activity, the external program may return codes of its own. Any values in the ReturnCode column other than 1, 0, and -1 are from the external program. See the external program's documentation for an explanation of its return codes.
Started	The date and time the activity started.	
Finished	The date and time the activity ended.	
Comment	Any comments associated with the activity.	

Downloading Flow History

You can download the information shown in the History page in Spectrum Management Console to a Microsoft Excel file.

1. Open Spectrum Management Console.
2. Go to **Flows** > **History**.
3. To download history information for services, click **Transaction History**. To download history for jobs and process flows, leave the **Flows** tab selected.
4. Click the Download button .

   Tip: If you want to download only certain entries in the history list, modify the filter settings to show only the history you want to download.

Purging Execution History

If you have many flows, or services that are used frequently, the execution history in Spectrum Management Console can become quite large. This procedure describes how to remove old records from the execution history. You may want to purge old records to reduce the size of the configuration database. Purging records before upgrading to a new version can help reduce the time it takes to upgrade Spectrum Technology Platform.

There are two purge methods:

- Purge records:

```
com.pb.spectrum.platform.configuration:manager=ArchiveTransactionManager
```

- Purge records and provide archive status:

```
com.pb.spectrum.platform.transaction:manager=archiveTransactionManager
```

The steps below demonstrate the "purge records" (non-archive status) request.

Note: To purge execution history for all nodes on a cluster, perform the purge on each node individually.

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. Under **Domain: com.pb.spectrum.platform.configuration**, click **com.pb.spectrum.platform.configuration:manager=ArchiveConfigurationManager**.
3. Optional: If you want to save an archive of the history that you are going to purge, specify the path where you want to save the archive in the **ArchiveDirectory** field, then click **set**. Then, set **ArchiveEnabled** to **true** and click **set**.
4. In the **ArchiveRetain** field, specify how many days of records you want to keep in the history then click **set**.
   For example, if you enter 45 then history records that are 45 days old or newer will be retained, and records 46 days old and older will be purged. To determine how many days of records you can retain, look at the job and process flow history in Spectrum Enterprise Designer and identify the point in time where the number of records exceeds 100,000.
5. Optional: If you want to schedule a purge to occur on a regular schedule, enter the schedule in the **CronExpression** field using a Cron expression.

   A cron expression consists of six space-separated values, with an optional seventh value:

   Seconds
   Minutes
   Hours
   Day of the month
   Month
   Day of the week
   Year (Optional)

   For example, this expression would purge the execution history at midnight every Sunday:

   ```
   0 0 0 ? * SUN
   ```

   For more information about cron expressions, see .

   After specifying a cron expression, click the **set** button next to the **CronExpression** field, set **PurgeEnabled** to **true**, and click the **set** button next to the **PurgeEnabled** field.

   > **Note:** You do not need to schedule purges if you want to purge the history only one time for the purposes of speeding up the upgrade process.

6. Optional: If you want to set a maximum number of records to remain in the history after the purge, specify the maximum number of records in the **MaxHistoryRecordCount** field.

   This is useful if you have a large number of history records each day, and even after purging old records based on the value in the **ArchiveRetain** field, the size of the execution history is still larger than you want. After purging the old records based on the value in the **ArchiveRetain** field, additional records will be purged until the number of records that remains is equal to the number in the **MaxHistoryRecordCount** field. If you do not want to specify a maximum number of history records, specify `-1`.

   > **Note:** The limit you specify in **MaxHistoryRecordCount** sets the limits for process flows and jobs separately. For example, if you specify 5000, the maximum number of process

flow history records will be 5,000, and the maximum number of job history records will be 5,000, for a total maximum number of 10,000 records.

7. In the **PurgeOperation** field, leave the value set to `ALL`.
8. Select **All MBeans** to return to the main JMX Console screen.
9. Under **Domain: com.pb.spectrum.platform.configuration**, select **com.pb.spectrum.platform.transaction:manager=ArchiveTransactionManager**.
10. To run the purge, click **Invoke**.

This purges the execution history so that you now have a smaller configuration database.

Triggering a Flow with a Control File

A flow can run automatically when a control file is detected in a monitored directory. This feature is useful in situations where the flow needs another process to complete before running. For example, you may have a flow that needs an input file generated by another business process. You can set up the other process to place a control file into a folder, and configure Spectrum Technology Platform to run a flow when that control file appears.

Note: Be sure that the control file is placed in the monitored folder only after all files required by the flow are in place and ready for processing.

1. If you have not already done so, expose the flow.

   You can expose a flow by opening it in Spectrum Enterprise Designer and selecting **File** > **Expose/Unexpose and Save**.

2. Open Spectrum Management Console.
3. Go to **Flows** > **Schedules**.
4. Click the Add button .
5. In the **Name** field, enter the name you want to give to this schedule. This is the name that will be displayed in the schedules listing.
6. In the **Flow** field, enter the job or process flow that you want to run. Only jobs and process flows that are saved and exposed are available here.
7. After you specify a flow, additional fields appear below the **Flow** field, one field for each of the flow's source stages (such as Read from File) and sink stages (such as Write to File).

   These fields show the files that will be used when the flow runs by this schedule. By default, the flow will use the files specified in the flow's sources and sinks. You can specify different files to use when this schedule runs by replacing the file path with the path to another file. For example, if your flow has a Read from File stage that reads data from `C:\FlowInput\Customers.csv`

but you want to use data from `C:\FlowInput\UpdatedCustomers.csv` when this schedule runs, you would specify `C:\FlowInput\UpdatedCustomers.csv` in the Read from File field.

> **Note:** In order change the files used in the source and sink stages you must have Read permission for the **Resources - File Servers** secured entity type.

Note that when a flow is triggered by a schedule the files used by a flow must reside on the Spectrum Technology Platform server or on a file server defined as an external resource in Spectrum Management Console. This applies both to jobs as well as job activities within a process flow. If a source or sink stage references a file on a client computer to modify the dataflow or override the dataflow file location.

Option 1: Move the file to the Spectrum Technology Platform server or file server then modify the dataflow:

a) Open the dataflow in Spectrum Enterprise Designer.
b) Double-click the source or sink stage.
c) In the **File name** field, click the browse button.
d) Click **Remote Machine** then select the file you want.

> **Note:** If you are running Spectrum Enterprise Designer on the same machine as the Spectrum Technology Platform server, it will appear that clicking Remote Machine is no different than clicking My Computer. However, you must select the file using Remote Machine in order for the system to recognize the file as being on the Spectrum Technology Platform server.

Option 2: Override the dataflow file location when this schedule runs.

You can override the file references contained in the flow when this schedule runs. To do this, replace the default file shown in each source and sink field with a path to a file on the Spectrum Technology Platform server or a file server resource defined in Spectrum Management Console.

8. In the **Trigger** field, choose **Control File**.
9. In the **Control file** field, specify the full path and name of the control file that will trigger the flow. You can specify an exact file name or you can use the asterisk (*) as a wild card. For example, `*.trg` would trigger the flow when any file with a `.trg` extension appears in the folder.

   The presence of a control file indicates that all files required for the flow are in place and ready to be used in the flow.

   The control file can be a blank file. For jobs, the control file can specify overrides to file paths configured in the Write to File or Read from File stages. To use a control file to override the file paths, specify the Read from File or Write from File stage names along with the input or output file as the last arguments like this:

```
stagename=filename
```

For example:

```
Read\ from\ File=file:C:/myfile_input.txt
Write\ to\ File=file:C:/myfile_output.txt
```

The stage name specified in the control file must match the stage label shown under the stage's icon in the flow. For example, if the input stage is labeled "Read From File" you would specify:

```
Read\ From\ File=file:C:/inputfile.txt
```

If the input stage is labeled "Illinois Customers" you would specify:

```
Illinois\ Customers=file:C:/inputfile.txt
```

When overriding a Read from File or Write to File location be sure to follow these guidelines:

- Start the path with the "file:" protocol. For example, on Windows specify "`file:C:/myfile.txt`" and on Linux specify "`file:/testfiles/myfile.txt`".
- The contents of the file must use an ASCII format ISO-8559-1 (Latin-1) compatible character encoding.
- You must use forward slashes in file paths, not backslashes.
- Spaces in stage names need to be escaped with a backslash.
- Stage names are case sensitive.

> **Note:** If there are multiple schedules that use a control file trigger, it is important that they each monitor different control files. Otherwise, the same control file may trigger multiple jobs or process flows causing unexpected behavior. For organizational purposes we recommend putting all required files and the control file in a dedicated directory.

10. In the **Poll interval** field, specify how frequently to check for the presence of the control file. For example, if you specify 10, the monitor will look every 10 seconds to see if the control file is in the monitored folder.

    The default is 60 seconds.

11. In the **Working folder** field, specify a folder where the control file will reside temporarily while the flow runs. Spectrum Technology Platform copies the file from the monitored folder to the working folder before running the flow. This clears out the monitored folder, which prevents the flow from being kicked off again by the same control file.
12. In the **Working folder options** field, specify what to do with the files in the working folder when the flow finishes running.

Keep	Leaves the files in their current location with their current name. If you select this option, the files in the working folder will be overwritten each time this schedule runs.
Move to	Moves the files from the working folder to a folder you specify. This allows you to preserve the files that were in the working folder by moving them to another location so that they are not overwritten the next time the file monitor

runs. You can also use this option to move the files to another monitored folder to trigger a downstream process, like another flow or some other process.

Rename with time stamp	Adds a time stamp to the file name in the working folder. This allows you to preserve a copy of the files in the working folder since the renamed file will have a unique name and so will not be overwritten the next time the monitor runs a flow.
Delete	Deletes the files from the working folder after the flow finishes running.

13. If the flow is configured to send email notifications you can specify additional recipients for the notifications that will be sent when this schedule runs. The recipients you specify here will receive notifications in addition to those recipients specified in the flow's notification settings. To configure a flow to send notifications, open the flow in Spectrum Enterprise Designer and go to **Edit** > **Notifications**.
14. Click **Save**.

Example: Monitored Folder and Working Folder

Let's say you have a car repair shop. Each day you want to mail the previous day's customers a coupon for a discount on future service. To accomplish this, you have a flow that takes the list of customers for the day, ensures the names have the correct casing, and validates the address. The list of customers for the day is generated by another system every evening. This other system generates a file containing the customer list, and you want to use this file as the input to the flow.

The system that generates the customer list puts it in a folder named `DailyCustomerReport`. It also places a blank trigger file in the folder when it is done. So you configure Spectrum Technology Platform to monitor this folder, specifying the trigger file as:

```
C:\DailyCustomerReport\*.trg
```

This tells Spectrum Technology Platform to run the flow whenever any file with a `.trg` extension appears in this folder. You could also specify a specific file name, but in this example we are using a wild card.

When a `.trg` file is detected in the `DailyCustomerReport` folder, Spectrum Technology Platform needs to move it to another folder before running the flow. The file must be moved because otherwise it would be detected again at the next polling interval, and this would result in the flow running again. So the file is moved to a "working folder" where it resides during the execution of the flow. You choose as the working folder `C:\SpectrumWorkingFolder`.

After the flow is finished processing the customer list, you want the trigger file to be moved to another location where it will trigger another process for billing. So, you select the **Move to** option and choose a folder named `C:\DailyBilling`.

So in this example, the trigger file starts off in `C:\DailyCustomerReport` and is then moved to the working folder `C:\SpectrumWorkingFolder`. After the flow is done, the trigger file is moved to `C:\DailyBilling` to initiate the billing process.

Command Line Execution

Running A Job from the Command Line

Before you can run a job from the command line, it must be exposed. To expose a job, open the job in Spectrum Enterprise Designer and select **File** > **Expose/Unexpose and Save**.

To run a job from the command line, you must install the job executor utility on the system where you want to run the job. The Job Executor is available from the Spectrum Technology Platform Welcome page on the Spectrum Technology Platform server (for example, http://myserver:8080).

Usage

`java -jar jobexecutor.jar -u` *UserID* `-p` *Password* `-j` *Job* `[Optional Arguments]`

Required	Argument	Description
No	`-?`	Prints usage information.
No	`-d` *delimiter*	Sets instance/status delimiter. This appears in synchronous output only.
No	`-e`	Use a secure HTTPS connection for communication with the Spectrum Technology Platform server.
No	`-f` *property file*	Specifies a path to a job property file. A job property file contains job executor arguments. For more information on job property files, see **Using a Job Property File** on page 200.
No	`-h` *host name*	Specifies the name or IP address of the Spectrum Technology Platform server.
No	`-i` *poll interval*	Specifies how often to check for completed jobs, in seconds. This applies only in synchronous mode.
Yes	`-j` *job name*	A comma-separated list of jobs to run. Job names are case-sensitive. Jobs are started in the order listed.

Required	Argument	Description
No	`-n` *email list*	Specifies a comma-separated list of additional email addresses for configured job notifications.

No	`-o` *property file*	Specifies a path to a flow options property file. Use a flow options property file to set options for stages in the flow. In order to set flow options using a property file, you must configure the flow to expose stage options at runtime. For more information, see **Adding Flow Runtime Options** on page 210.
		
		For example, a flow options properties file for a flow that contains an Assign GeoTAX Info stage may look like this:
		
		`OutputCasing=U`
		`UseStreetLevelMatching=N`
		`TaxKey=T`
		`Database.GTX=gsl`

Yes	`-p` *password*	The password of the user.
No	`-r`	Specify this argument to return a detailed report about the job. This option only works if you also specify `-w` . The report contains this information: • **Position 1** - Name of job • **Position 2** - Job process ID • **Position 3** - Status • **Position 4** - Start Date-Time (MM/DD/YYYY HH:MM:SS) • **Position 5** - End Date-Time (MM/DD/YYYY HH:MM:SS) • **Position 6** - Number of successful records • **Position 7** - Number of failed records • **Position 8** - Number of malformed records • **Position 9** - Currently unused For example: `MySimpleJob|4|succeeded|04/09/2019 14:50:47|04/09/2019 14:50:47|100|0|0|` The information is delimited using the delimiter specified in the `-d` argument.
No	`-s` *port*	The socket (port) on which the Spectrum Technology Platform server is running. The default is 8080.
No	`-t` *timeout*	Sets the timeout (in seconds) for synchronous mode. The default is 3600. The maximum is 2147483. This is a global, aggregate timeout and represents the maximum time to wait for all spawned jobs to complete.

Required	Argument	Description
Yes	`-u` *user name*	The login name of the user.
No	`-v`	Return verbose output.
No	`-w`	Runs job executor in synchronous mode. This means that job executor remains running until the job completes. If you do not specify `-w`, job executor exits after starting the job, unless the job reads from or writes to files on the server. In this case, job executor will run until all local files are processed, then exit.
No	*StageName*	Overrides the input or output file specified in Read from File or Write to File. This is defined in the format: *StageName=Protocol:FileName* For more information, see **Overriding Job File Locations** on page 195.
No	*StageName*:schema	Overrides the file layout definition specified in Read from File or Write to File with one defined in a schema file. This is defined in the format: *StageName*:schema=*Protocol*:*SchemaFile* For more information, see **Overriding the File Format at the Command Line** on page 198.

Example Use of Job Executor

This example shows command line invocation and output:

```
D:\spectrum\job-executor>java -jar jobexecutor.jar -u user123
 -p "mypassword" -j validateAddressJob1 -h
spectrum.example.com -s 8888 -w -d "%" -i 1 -t 9999

validateAddressJob1%105%succeeded
```

In this example, the output indicates that the job named 'validateAddressJob1' ran (with identifier 105) with no errors. Other possible results include "failed" or "running."

Overriding Job File Locations

When you run a job at the command line using job executor or the Administration Utility, you can override the input file specified in the flow's source stage (such as Read from File), as well as the output file specified in the flow's sink stage (such as Write to File).

To do this in job executor, specify this command at the end of the job executor command:

StageName=Protocol:FileName

In the Administration Utility, use the `--l` argument in the `job execute` command:

`--l` *StageName=Protocol:FileName*

Where:

StageName

The stage label shown under the stage's icon in the flow in Spectrum Enterprise Designer. For example, if the stage is labeled "Read from File" you would specify `Read from File` for the stage name.

To specify a stage within an embedded flow or a subflow, preface the stage name with the name of the embedded flow or subflow, followed by a period then the stage name:

EmbeddedOrSubflowName.StageName

For example, to specify a stage named Write to File in a subflow named Subflow1, you would specify:

```
Subflow1.Write to File
```

To specify a stage in an embedded flow that is within another embedded flow, add the parent flow, separating each with a period. For example, if Embedded Dataflow 2 is inside Embedded Dataflow 1, and you want to specify the Write to File stage in Embedded Dataflow 2, you would specify this:

```
Embedded Dataflow 1.Embedded Dataflow 2.Write to File
```

Protocol

A communication protocol that can be one of these types:

file Use the file protocol if the file is on the same machine as the Spectrum Technology Platform server. For example, on Windows specify:

```
"file:/C:/myfile.txt"
```

On Linux specify:

```
"file:/testfiles/myfile.txt"
```

esclient Use the esclient protocol if the file is on the computer where you are executing the job if it is a different computer from the one running the Spectrum Technology Platform server. Use this format:

esclient:*ComputerName/path to file*

For example,

```
esclient:mycomputer/testfiles/myfile.txt
```

Note: If you are executing the job on the server itself, you can use either the file or esclient protocol, but are likely to have better performance using the file protocol.

If the host name of the Spectrum Technology Platform server cannot be resolved, you may get the error "Error occurred accessing file". To resolve this issue, open this file on the server: *SpectrumDirectory*`/server/conf/spectrum-container.properties`. Set the `spectrum.runtime.hostname` property to the IP address of the server.

esfile Use the esfile protocol if the file is on a file server. The file server must be defined in Spectrum Management Console as a resource. Use this format:

`esfile://`*`file server/path to file`*

For example,

```
esfile://myserver/testfiles/myfile.txt
```

Where myserver is an FTP file server resource defined in Spectrum Management Console.

webhdfs Use the webhdfs protocol if the file is on a Hadoop Distributed File Server. The HDFS server must be defined in Spectrum Management Console as a resource. Use this format:

`webhdfs://`*`file server/path to file`*

For example,

```
webhdfs://myserver/testfiles/myfile.txt
```

Where myserver is an HDFS file server resource defined in Spectrum Management Console.

FileName

The full path to the file you want to use as input or output.

Note: You must use forward slashes in file paths. Do not use backslashes.

To specify multiple overrides, separate each override with a comma.

Example File Override

The required job executor command would use the file `C:/myfile_input.txt` as the input file for the Read from File stage and would use the file `C:/myfile_output.txt` as the output file for the Write to File stage.

```
java -jar jobexecutor.jar -j Job1 -u Bob1234 -p "" "Read from
File"="file:/C:/myfile_input.txt" "Write to
File"="file:/C:/myfile_output.txt"
```

Overriding the File Format at the Command Line

When you run a job using job executor or the Administration Utility, you can override the file layout (or schema) of the file specified in the flow's Read from File stage and Write to File stage.

To do this in job executor, specify this at the end of the job executor command line command:

```
StageName:schema=Protocol:SchemaFile
```

In the Administration Utility, use the `--l` argument in the `job execute` command:

```
--l StageName:schema=Protocol:SchemaFile
```

Where:

StageName

The stage label shown under the stage's icon in the flow in Spectrum Enterprise Designer. For example, if the stage is labeled "Read from File" you would specify `Read from File` for the stage name.

To specify a stage within an embedded flow or a subflow, preface the stage name with the name of the embedded flow or subflow, followed by a period then the stage name:

```
EmbeddedOrSubflowName.StageName
```

For example, to specify a stage named Write to File in a subflow named Subflow1, you would specify:

```
Subflow1.Write to File
```

To specify a stage in an embedded flow that is within another embedded flow, add the parent flow, separating each with a period. For example, if Embedded Dataflow 2 is inside Embedded Dataflow 1, and you want to specify the Write to File stage in Embedded Dataflow 2, you would specify this:

```
Embedded Dataflow 1.Embedded Dataflow 2.Write to File
```

Protocol

A communication protocol:

file Use the file protocol if the file is on the same machine as the Spectrum Technology Platform server. For example, on Windows specify:

```
"file:/C:/myfile.txt"
```

On Linux specify:

```
"file:/testfiles/myfile.txt"
```

esclient Use the esclient protocol if the file is on the computer where you are executing the job if it is a different computer from the one running the Spectrum Technology Platform server. Use this format:

```
esclient:ComputerName/path to file
```

For example,

```
esclient:mycomputer/testfiles/myfile.txt
```

> **Note:** If you are executing the job on the server itself, you can use either the file or esclient protocol, but are likely to have better performance using the file protocol.

If the host name of the Spectrum Technology Platform server cannot be resolved, you may get the error "Error occurred accessing file". To resolve this issue, open this file on the server: *SpectrumDirectory*`/server/conf/spectrum-container.properties`. Set the `spectrum.runtime.hostname` property to the IP address of the server.

esfile Use the esfile protocol if the file is on a file server. The file server must be defined in Spectrum Management Console as a resource. Use this format:

```
esfile://file server/path to file
```

For example,

```
esfile://myserver/testfiles/myfile.txt
```

Where myserver is an FTP file server resource defined in Spectrum Management Console.

webhdfs Use the webhdfs protocol if the file is on a Hadoop Distributed File Server. The HDFS server must be defined in Spectrum Management Console as a resource. Use this format:

```
webhdfs://file server/path to file
```

For example,

```
webhdfs://myserver/testfiles/myfile.txt
```

Where myserver is an HDFS file server resource defined in Spectrum Management Console.

SchemaFile

The full path to the file that defines the layout you want to use.

> **Note:** You must use forward slashes in file paths. Do not use backslashes.

To create a schema file, define the layout you want in Read from File or Write to File, then click the **Export** button to create an XML file that defines the layout.

Note: You cannot override a field's data type in a schema file when using job executor. The value in the <Type> element, which is a child of the FieldSchema element, must match the field's type specified in the flow's Read from File or Write to File stage.

Example File Format Override

The job executor command below uses the file `C:/myschema.xml` as the layout definition for the file read in by the Read from File stage.

```
java -jar jobexecutor.jar -j Job1 -u Bob1234 -p "" "Read from File":schema="file:/C:/myschema.xml"
```

Using a Job Property File

A job property file contains arguments that control the execution of jobs when you use the job executor or the Administration Utility to run a job. Use a job property file if you want to reuse arguments by specifying a single argument at the command line (`-f`) rather than specifying each argument individually at the command line.

To create a property file, create a text file with one argument on each line.

```
d %
h spectrum.mydomain.com
i 30
j validateAddressJob1
u user
p password
s 8888
t 9999
w true
```

The job property file can contain these arguments:

Required	Argument	Description
No	`?`	Prints usage information.
No	`d` *delimiter*	Sets instance/status delimiter. This appears in synchronous output only.
No	`e`	Use a secure HTTPS connection for communication with the Spectrum Technology Platform server.
No	`h` *hostname*	Specifies the name or IP address of the Spectrum Technology Platform server.
No	`i` *pollinterval*	Specifies how often to check for completed jobs, in seconds. This applies only in synchronous mode.
Yes	`j` *jobname*	A comma-separated list of jobs to run. Job names are case-sensitive. Jobs are started in the order listed.

Required	Argument	Description
No	`n` *emaillist*	Specifies a comma-separated list of additional email addresses for configured job notifications.
Yes	`p` *password*	The password of the user.
No	`r`	Returns a delimited list with this information about the job written to standard output: • **Position 1** - Name of job • **Position 2** - Job process ID • **Position 3** - Status • **Position 4** - Start Date - Time (MM/DD/YYYY HH:MM:SS) • **Position 5** - End Date - Time (MM/DD/YYYY HH:MM:SS) • **Position 6** - Number of successful records • **Position 7** - Number of failed records • **Position 8** - Number of malformed records • **Position 9** - Currently unused The information is delimited using the delimiter specified in the -d argument. For example: `MySimpleJob|4|succeeded|04/09/2019 14:50:47|04/09/2019 14:50:47|100|0|0|`
No	`s` *port*	The socket (port) on which the Spectrum Technology Platform server is running. The default is 8080.
No	`t` *timeout*	Sets the timeout (in seconds) for synchronous mode. The default is 3600. The maximum is 2147483. This is a global, aggregate timeout and represents the maximum time to wait for all spawned jobs to complete.
Yes	`u` *username*	The login name of the user.
No	`v`	Return verbose output.
No	`w`	Specifies to wait for jobs to complete in a synchronous mode.

Using Both Command Line Arguments and a Property File

A combination of both command-line entry and property file entry is also valid. For example:

```
java -jar jobexecutor.jar -f /dcg/job.properties -j job1
```

In this case command line arguments take precedence over arguments specified in the properties file. In the above example, the job job1 would take precedence over a job specified in the properties file.

Overriding Input and Output Files Using a Job Property File

You can override the input file specified in the dataflow source stage (such as Read from File), as well as the output file specified in the dataflow sink stage (such as Write to File) in a job executor property file. To do this, specify the following in the property file:

```
StageName\:file=Protocol:FileName
```

Where:

StageName

The stage label shown under the stage's icon in the dataflow in Spectrum Enterprise Designer. Use a backslash before any spaces, colons, or equal signs in the stage name. For example, if the stage is labeled "Read from File" you would specify `Read\ from\ File` for the stage name.

```
Embedded\ Dataflow\ 1.Embedded\ Dataflow\ 2.Write\ to\ File
```

To specify a stage within an embedded dataflow or a subflow, preface the stage name with the name of the embedded dataflow or subflow, followed by a period then the stage name:

```
EmbeddedOrSubflowName.StageName
```

For example, to specify a stage named Write to File in a subflow named Subflow1, you would specify:

```
Subflow1.Write\ to\ File
```

To specify a stage in an embedded dataflow that is within another embedded dataflow, add the parent dataflow, separating each with a period. For example, if Embedded Dataflow 2 is inside Embedded Dataflow 1, and you want to specify the Write to File stage in Embedded Dataflow 2, you would specify this:

```
Embedded\ Dataflow\ 1.Embedded\ Dataflow\ 2.Write\ to\ File
```

Note: You must include `:file` after the stage name. For example, `Read\ from\ File:file`. This is different from the syntax used to override files at the command line where `:file` is not specified after the stage name.

Protocol

A communication protocol. One of the following:

file Use the file protocol if the file is on the same machine as the Spectrum Technology Platform server. For example, on Windows specify:

```
"file:/C:/myfile.txt"
```

On Linux specify:

```
"file:/testfiles/myfile.txt"
```

esclient Use the esclient protocol if the file is on the computer where you are executing the job if it is a different computer from the one running the Spectrum Technology Platform server. Use this format:

```
esclient:ComputerName/path to file
```

For example,

```
esclient:mycomputer/testfiles/myfile.txt
```

> **Note:** If you are executing the job on the server itself, you can use either the file or esclient protocol, but are likely to have better performance using the file protocol.

If the host name of the Spectrum Technology Platform server cannot be resolved, you may get the error "Error occurred accessing file". To resolve this issue, open this file on the server: *SpectrumDirectory*`/server/conf/spectrum-container.properties`. Set the `spectrum.runtime.hostname` property to the IP address of the server.

esfile Use the esfile protocol if the file is on a file server. The file server must be defined in Spectrum Management Console as a resource. Use this format:

```
esfile://file server/path to file
```

For example,

```
esfile://myserver/testfiles/myfile.txt
```

Where myserver is an FTP file server resource defined in Spectrum Management Console.

webhdfs Use the webhdfs protocol if the file is on a Hadoop Distributed File Server. The HDFS server must be defined in Spectrum Management Console as a resource. Use this format:

```
webhdfs://file server/path to file
```

For example,

```
webhdfs://myserver/testfiles/myfile.txt
```

Where myserver is an HDFS file server resource defined in Spectrum Management Console.

Example

The last two lines of the following property file specify the files for the Read from File stage and the Write to File stage.

```
j=testJob
h=myspectrumserver.example.com
s=8080
u=david1234
```

```
p=mypassword1234
Read\ from\ File\:file=file:C:/myfile_input.txt
Write\ to\ File\:file=file:C:/myfile_output.txt
```

Overriding File Format Using a Job Property File

You can use a property file to override the file layout (or schema) of the file specified in the dataflow Read from File stage and Write to File stage. To do this, specify the following in the property file:

```
StageName\:schema=Protocol:SchemaFile
```

Where:

StageName

The stage label shown under the stage's icon in the dataflow in Spectrum Enterprise Designer. Use a backslash before any spaces, colons, or equal signs in the stage name. For example, if the stage is labeled "Read from File" you would specify `Read\ from\ File` for the stage name.

```
Embedded\ Dataflow\ 1.Embedded\ Dataflow\ 2.Write\ to\ File
```

To specify a stage within an embedded dataflow or a subflow, preface the stage name with the name of the embedded dataflow or subflow, followed by a period then the stage name:

```
EmbeddedOrSubflowName.StageName
```

For example, to specify a stage named Write to File in a subflow named Subflow1, you would specify:

```
Subflow1.Write\ to\ File
```

To specify a stage in an embedded dataflow that is within another embedded dataflow, add the parent dataflow, separating each with a period. For example, if Embedded Dataflow 2 is inside Embedded Dataflow 1, and you want to specify the Write to File stage in Embedded Dataflow 2, you would specify this:

```
Embedded\ Dataflow\ 1.Embedded\ Dataflow\ 2.Write\ to\ File
```

Note: You must include `:file` after the stage name. For example, `Read\ from\ File:file`. This is different from the syntax used to override files at the command line where `:file` is not specified after the stage name.

Protocol

A communication protocol. One of the following:

file Use the file protocol if the file is on the same machine as the Spectrum Technology Platform server. For example, on Windows specify:

```
"file:/C:/myfile.txt"
```

On Linux specify:

```
"file:/testfiles/myfile.txt"
```

esclient Use the esclient protocol if the file is on the computer where you are executing the job if it is a different computer from the one running the Spectrum Technology Platform server. Use this format:

```
esclient:ComputerName/path to file
```

For example,

```
esclient:mycomputer/testfiles/myfile.txt
```

> **Note:** If you are executing the job on the server itself, you can use either the file or esclient protocol, but are likely to have better performance using the file protocol.

If the host name of the Spectrum Technology Platform server cannot be resolved, you may get the error "Error occurred accessing file". To resolve this issue, open this file on the server: *SpectrumDirectory*`/server/conf/spectrum-container.properties`. Set the `spectrum.runtime.hostname` property to the IP address of the server.

esfile Use the esfile protocol if the file is on a file server. The file server must be defined in Spectrum Management Console as a resource. Use this format:

```
esfile://file server/path to file
```

For example,

```
esfile://myserver/testfiles/myfile.txt
```

Where myserver is an FTP file server resource defined in Spectrum Management Console.

webhdfs Use the webhdfs protocol if the file is on a Hadoop Distributed File Server. The HDFS server must be defined in Spectrum Management Console as a resource. Use this format:

```
webhdfs://file server/path to file
```

For example,

```
webhdfs://myserver/testfiles/myfile.txt
```

Where myserver is an HDFS file server resource defined in Spectrum Management Console.

SchemaFile

The full path to the file that defines the layout you want to use.

Note: You must use forward slashes in file paths. Do not use backslashes.

To create a schema file, define the layout you want in Read from File or Write to File, then click the **Export** button to create an XML file that defines the layout.

Note: You cannot override a field's data type in a schema file when using job executor. The value in the <Type> element, which is a child of the FieldSchema element, must match the field's type specified in the flow's Read from File or Write to File stage.

Example

In the following example properties file, the last line overrides the file layout defined in the Read from File stage with the layout defined in the file `inputSchema.xml`. A backslash is used before the spaces in the stage's name.

```
j=testJob
h=myspectrumserver.example.com
s=8080
u=david1234
p=mypassword1234
Read\ from\ File\:file=esclient:c:/MyData/testInput.txt
Read\ from\ File\:schema=esclient:c:/MyData/inputSchema.xml
```

Running a Process Flow from the Command Line

To run a process flow from the command line, use the Process Flow Executor. You can install the Process Flow Executor from the Spectrum Technology Platform Welcome page (for example, http://myserver:8080).

Note: You can also use the Administration Utility to run process flows from the command line.

Usage

`java -jar pflowexecutor.jar -r` *ProcessFlowName* `-u` *UserID* `-p` *Password* `[Optional Arguments]`

Required	Argument	Description
No	`-?`	Prints usage information.
No	`-d` *DelimiterCharacter*	Sets a delimiter to use to separate the status information displayed in the command line when you run the command. The default is "|". For example, using the default character, the message below is

Required	Argument	Description
		displayed at the command line when you run a process flow named "MyProcessflow": `MyProcessflow|1|Succeeded`
No	`-e`	Use an HTTPS connection for communication with the Spectrum Technology Platform server. **Note:** If you specify any file overrides this argument must not be the last argument specified.
No	`-f` *PropertyFile*	Specifies a path to a property file. For more information on property files, see **Using a Process Flow Property File** on page 208.
No	`-h` *HostName*	Specifies the name or IP address of the Spectrum Technology Platform server.
No	`-i` *PollInterval*	Specifies how often to check for completed jobs, in seconds. The default is "5".
Yes	`-p` *Password*	The password of the user. Required.
Yes	`-r` *ProcessFlowNames*	A comma-separated list of process flows to run. Required. **Note:** If you specify any file overrides this argument must not be the last argument specified.
No	`-s` *Port*	The socket (port) on which the Spectrum Technology Platform server is running. The default is 8080.
No	`-t` *Timeout*	This option is deprecated and will be ignored.
Yes	`-u` *UserName*	The login name of the user. Required.
No	`-v` *Verbose*	Return verbose output where *Verbose* is one of the following: **true** Return verbose output. **false** Do not return verbose output. **Note:** If you specify any file overrides this argument must not be the last argument specified.
No	`-w` *WaitToComplete*	This option is deprecated and will be ignored.
No	*StageName=FileName*	Overrides the input or output file specified in the job. For more information, see **Overriding Process Flow File Locations** on page 209.

Examples

This is a basic command-line entry, with a process flow name and user ID, and password:

```
java -jar pflowexecutor.jar -r MyFlow1 -u Bob1234 -p "mypassword1"
```

This example shows the same information as above but with additional arguments:

```
java -jar pflowexecutor.jar -r Flow1 -u Bob1234 -p "mypassword1" -h spectrum.example.com -s 8080 -w -d
 "%" -i 1
```

This example shows command line invocation and output.

```
D:\spectrum\pflow-executor>java -jar pflowexecutor.jar -u Bob1234 -p
"mypassword1" -r
validateAddressFlow1 -h spectrum.example.com -s 8080 -w -d "%" -i
1 -t 9999
validateAddressJob1%111%succeeded
```

In this example, the process flow named validateAddressFlow1 ran (with identifier 111). No errors occurred. Other possible results include "failed" or "running."

Using a Process Flow Property File

A property file contains arguments that you can reuse by specifying the path to the property file with the `-f` argument in the process flow executor. The property file must contain, at minimum, the process flow (`r`), user ID (`u`), and password (`p`).

1. Open a text editor.
2. Specify one argument on each line as shown in the example below. See **Running a Process Flow from the Command Line** on page 206 for a list of arguments.

   Note: You cannot use a property file to override input or output files. Overriding input and output files can only be done using command line arguments.

   ```
   d=%
   h=myserver.mydomain.com
   i=30
   u=user
   p=password
   r=MyFlow1
   s=8888
   ```

3. Save the file with a file extension of `*.properties` (for example, `example.properties`).
4. When you run the process flow executor, specify the path to the property file using the `-f` argument. A combination of both command-line entry and property file entry is also valid. Command line arguments take precedence over arguments specified in the properties file.
   `java -jar pflowexecutor.jar -f /dcg/flow.properties -r MyFlow2`

In the above example, the process flow MyFlow2 would take precedence over a process flow specified in the properties file.

Overriding Process Flow File Locations

When you run a process flow using the Process Flow Executor command line tool, you can specify that the process flow should use different input and output files than those specified in the job referenced by the process flow. To do this, specify the Read from File or Write from File stage names along with the input or output file as the last arguments like this:

```
"<jobname>|<stagename>"="<filename>"
```

Where:

JobName

The name of a job referenced in the process flow.

StageName

The name of a Read from File or Write to File stage in the job as shown in the stage label under the stage's icon in the dataflow. For example, if the input stage is labeled "Read From File" you would specify:

```
"Job1|Read From File"="file:C:/inputfile.txt"
```

If the input stage is labeled "Illinois Customers" you would specify:

```
"Job1|Illinois Customers"="file:C:/inputfile.txt"
```

File

The protocol and full path to the file. You must use forward slashes in file paths, not backslashes. The protocol must be one of these:
file:
If the file is on the same machine as the Spectrum Technology Platform server, start the path with the "file:" protocol. For example, on Windows specify `file:C:/myfile.txt` and on Linux specify `file:/testfiles/myfile.txt`.

> **Note:** If the client and server are running on the same machine, you can use either the "file:" or "esclient:" protocol, but are likely to have get better performance using the "file:" protocol.

esclient:
If the file is on the same machine as Process Flow Executor, start the path with the "esclient:" protocol. For example, on Windows specify `esclient:C:/myfile.txt` and on Linux specify `esclient:/testfiles/myfile.txt`.

> **Note:** If the machine running process flow executor cannot resolve the host name of the Spectrum Technology Platform server, you may get an error "Error occurred accessing file". To resolve this issue, open this file on the server:

SpectrumDirectory`/server/conf/spectrum-container.properties`. Set the `spectrum.runtime.hostname` property to the IP address of the server.

ftp:

To use a file server defined in Spectrum Management Console, use this format: `ftp:`*`NameOfFileServer`*`/`*`PathToFile`*. For example, `ftp://FS/testfiles/myfile.txt` where FS is a file server resource defined in Spectrum Management Console.

For example,

```
java -jar pflowexecutor.jar -r Flow1 -u Bob1234 -p "mypassword1" -h
spectrum.example.com -s 8080 -w -d "%" -i 1 "Job1|Read from
File"="file:C:/myfile_input.txt" "Job1|Write to
File"="file:C:/myfile_output.txt"
```

Adding Flow Runtime Options

Flow runtime options enable you control the behavior of stages when you run the flow. This is useful when you want to have the ability to modify the behavior of the flow when it runs. For example, you may want to specify a source database for a Read from DB stage when you run the flow, rather than using the database specified in the Read from DB stage in the flow.

This procedure describes how to expose options that can be set at runtime. After performing this procedure you will be able to set flow options at runtime using these techniques:

- For jobs, you will be able to specify runtime options using a flow options property file and job executor's `-o` argument.
- For services, you will be able to specify runtime options as API options.
- For services exposed as web service, you will be able to specify runtime options as parameters in the request.
- For subflows, runtime options will be inherited by the parent flow and exposed through one of the above means, depending on the parent flow type (job, service, or service exposed as a web service).

To add runtime options to a flow,

1. Open the flow in Spectrum Enterprise Designer.
2. If you want to configure runtime options for a stage in an embedded flow, open the embedded flow.
3. Click the Dataflow Options icon on the toolbar or click **Edit** > **Dataflow Options**. The **Dataflow Options** dialog box appears.
4. Click **Add**. The **Define Dataflow Options** dialog box appears.

5. In the **Option name** field, specify the name you want to use for this option. This is the option name that will have to be specified at runtime in order to set this option.
6. In the **Label** field, you can specify a different label or keep it the same as the option name.
7. Enter a description of the option in the **Description** field.
8. In the **Target** field, chose whether you want this option to be applied to all stages in the flow or only certain stages.

   Selected stage(s)

   Select this option if you want the option to only be applied to the stages you specify.

   All stages

   Select this option if you want the option to be applied to all stages in the flow.

   Includes transforms

   Select this option if you want the runtime option to be made available to custom transforms in Transformer stages in the flow. If you choose this option you can access the value specified when you run it in the Groovy script using this syntax:

   ```
   options.get("optionName")
   ```

   For example, to access an option named `casing`, you would include this in your custom transform script:

   ```
   options.get("casing")
   ```

9. If you chose **Selected stage(s)** in the **Target** field, the **Map dataflow options to stages** table displays a list of the stages in the flow. Select the option that you want to expose as a flow option. You will see the **Default value** and **Legal values** fields be completed with data when you select your first item.

   Note: You can select multiple options so that the flow option can control multiple stages options. If you do this, each of the stage options you select must share legal values. For example, one option has values of Y and N, each of the additional options must have either Y or N in their set of values, and you can only allow the value in common to be available at runtime. So, if you select an option with Y and N values, you cannot select an option with the values of E, T, M, and L, but you could select an option with the values of P, S, and N because both options share "N" as a value. However, only "N" would be an available value for this option, not "Y", "P", or "S".

10. If you want to limit the values that can be specified at runtime, edit the options in the **Legal values** field by clicking on the icon just to the right of the field.
11. If you want to change the default value, specify a different value in the **Default value** field.

    Note: For a service, you can only modify default values before exposing the service for the first time. Once you expose the service you can no longer modify default values using Spectrum Enterprise Designer. Instead, you must use Spectrum Management Console. For more information, see **Specifying Default Service Options** on page 157.

12. Click **OK**.
13. Continue adding options as desired.
14. Click **OK** in the **Dataflow Options** dialog box when you are done adding options.
15. If you added a runtime option to an embedded flow, you must define the runtime option parent flow as well as all ancestor flows in order to make the options available at runtime. To do this, open the flow that contains the embedded flow and expose the option you just created. If necessary, open the parent of that flow and define the option there, and so on until all ancestors have the flow option defined.

    For example, say you had a flow named "A" that contained an embedded flow named "B" which contained an embedded flow named "C", so that you have an embedded flow hierarchy like this: A > B > C. If you wanted to expose an option named Casing in a stage in embedded flow "C", you would open embedded flow "C" and define it. Then, you would open embedded flow "B" and define the option. Finally, you would open flow "A" and define the option, making it available at runtime.

The flow is now configured to allow options to be specified at runtime.

8 - Performance

In this section

Performance Tuning Checklist.....214
Monitoring Performance.....228

Performance Tuning Checklist

This checklist describes the approach we recommend for getting optimal performance from your Spectrum Technology Platform environment. The techniques are listed in order from those having the most significant impact on performance to those having the least.

Performance Setting	Description	Additional Information
Dataflow design	The most significant impact on performance is the design of the dataflow. There are several best practices you can follow to ensure that your dataflows have good performance.	**Design guidelines for optimal performance** on page 215
Dataflow pool size	Dataflow pool size controls how may instances of a service dataflow can be running at a time. The default pool size for a dataflow is eight.	**Dataflow Pool Size** on page 216
Database pool size and stage runtime instances	Database pool size and stage runtime instances control the ability of the system to handle multiple requests concurrently. They need to be adjusted in tandem.	**Database Pool Size and Runtime Instances** on page 150
Sort performance	Sorting large data sets can be one of the most time-consuming operations performed during batch processing. Sort performance options control memory and disk utilization, allowing you to take full advantage of the available memory and disk capacity.	**Setting Default Sort Performance Options** on page 219
Remote component options	These settings control memory usage for certain stages such as address validation and geocoding stages.	**Configuring Remote Component Options** on page 221
Individual stages	There are some actions you can take to optimize specific types of processing. Review the best practices for performance to make sure that you have configured your dataflow to achieve optimal performance.	**Optimizing Individual Stages** on page 222
JVM settings	There are JVM options that will yield performance gains with certain hardware.	**JVM Performance Tuning** on page 227
Micro-batch processing	Micro-batch processing is a technique where you include more than one record in a single service request.	**Micro-batch Processing** on page 227

Performance Setting	Description	Additional Information
Heap size configuration for Elasticsearch	Increase the Elasticsearch heap size if you are performing memory-intensive operations with the profiling tool in Discovery or with any of the matching stages in a flow.	**Heap Size Configuration for Elasticsearch** on page 221

Design guidelines for optimal performance

Carefully designing your flows to optimize performance is the most important thing you can do to achieve good performance on Spectrum Technology Platform. These guidelines describe techniques you can use optimize flow performance.

Minimize the Number of Stages

Spectrum Technology Platform achieves high performance through parallel processing. Each stage in a flow runs asynchronously in its own thread. However, it is possible to overthread the processors when executing certain types of flows. When this happens, the system spends as much or more time managing threads as doing "real work". We have seen flows with as many as 130 individual stages that perform very poorly on smaller servers with one or two processors.

So the first consideration in designing flows that perform well is to use as many stages as needed, but no more. Some examples of using more stages than needed are:

- Using multiple conditional routers where one would suffice
- Defining multiple transformer stages instead of combining the transforms in a single stage

Fortunately it is usually possible to redesign these flows to remove redundant or unneeded stages and improve performance.

For complex flows, consider using embedded flows or subflows to reduce clutter on the canvas and make it easier to view and navigate the flow. Using embedded flows does not have a performance benefit at runtime, but it does make it easier to work with flows in Spectrum Enterprise Designer. Using subflows to simplify complex flows can improve Spectrum Enterprise Designer performance when editing flows.

Reduce Record Length

Since data is being passed between concurrently executing stages, another consideration is the length of the input records. Generally input with a longer record length will take longer to process than input with a shorter record length, simply because there is more data to read, write, and sort. Dataflows with multiple sort operations will particularly benefit from a reduced record length. In the case of very large record lengths it can be faster to remove the unnecessary fields from the input

prior to running the Spectrum Technology Platform job then append them back to the resulting output file.

Use Sorting Appropriately

Another consideration is to minimize sort operations. Sorting is often more time consuming than other operations, and can become problematic as the number and size of input records increases. However, many Spectrum Technology Platform stages either require or prefer sorted input data. Spectrum Universal Addressing and Enterprise Geocoding, for example, perform optimally when the input is sorted by country and postal code. Stages such as Intraflow Match and Interflow Match require that the input be sorted by the "group by" field. In some cases you can use an external sort application to presort the input data and this can be faster than sorting within the Spectrum Technology Platform flow.

Dataflow Pool Size

Dataflow pool size controls how may instances of each service dataflow can be running at a time. You can increase the pool size to improve performance to a point, but increased pool size may result in reduced performance if the server does not have the processor or memory resources available to handle several instances of each service dataflow running concurrently. If processor and memory resources are being used to their limit, you may find that reducing dataflow pool size, which limits the number of concurrent instances of each service dataflow, may provide more acceptable performance overall.

When finding the right pool size for your system, keep in mind that the dataflow pool size limits the number of instances of each service dataflow, not the total number of concurrent service dataflows. For example, with the default setting of 8, each service dataflow is allowed to have eight instances running at a time. If there are two service dataflows each utilizing the maximum of 8 concurrent instances, there would be 16 total instances of service dataflows running concurrently on your system.

Note: Dataflow pool size affects the performance of services only, not jobs.

To configure the dataflow pool size, see **Monitoring Performance with the Spectrum JMX Console** on page 231.

Database Pool Size and Runtime Instances

In most Spectrum Technology Platform environments there are multiple flows running at the same time, whether they are batch jobs or services responding to web service or API requests. To optimize concurrent processing, you can use the database pool size setting, which limits the number of concurrent requests a Spectrum database handles, and runtime instances, which controls the number

of instances of a flow stage that run concurrently. These two settings should be tuned together to achieve optimal performance.

Database Pool Size

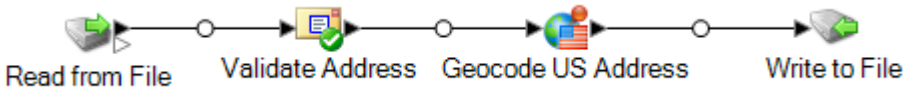
Spectrum databases contain reference data used by certain stages, such as postal data used to validate addresses, or geocoding data used to geocode addresses. These databases can be configured to accept multiple concurrent requests from the stages or services that use them, thereby improving the performance of the flows or service requests. The database pool size sets the maximum number of concurrent requests that a Spectrum database will process. By default, Spectrum databases have a pool size of 4, meaning the database can process four requests simultaneously.

The optimal pool size varies by module. You will generally see the best results by setting the pool size between one-half to twice the number of CPUs on the server, with the optimal pool size for most modules being the same as the number of CPUs. For example, if your server has four CPUs you may want to experiment with a pool size between 2 (one-half the number of CPUs) and 8 (twice the number of CPUs) with the optimal size possibly being 4 (the number of CPUs).

When modifying the pool size you must also consider the number of runtime instances specified in the flow for the stages accessing the database. Consider for example a flow that has a Geocode US Address stage that is configured to use one runtime instance. If you set the pool size for the US geocoding database to four, you will not see a performance improvement because there would be only one runtime instance and therefore there would only be one request at a time to the database. However, if you were to increase the number of runtime instances of Geocode US Address to four, you might then see an improvement in performance since there would be four instances of Geocode US Address accessing the database simultaneously, therefore using the full pool.

Runtime Instances

Each stage in a flow operates asynchronously in its own thread and is independent of any other stage. This provides for parallel processing of stages in a flow, allowing you to utilize more than one runtime instance for a stage. This is useful in flows where some stages process data faster than others. This can lead to an unbalanced distribution of work among the threads. For example, consider a flow consisting of these stages:



Depending on the configuration of the stages, it may be that the Validate Address stage processes records faster than the Geocode US Address stage. If this is the case, at some point during the execution of the flow all the records will have been processed by Validate Address, but Geocode US Address will still have records to process. In order to improve performance of this flow, it is necessary to improve the performance of the slowest stage - in this case Geocode US Address. One way to do that is to specify multiple runtime instances of the stage. Setting the number of runtime instances to two, for example, means that there will be two instances of that stage, each running in its own thread, available to process records. Keep in mind that while specifying multiple runtime

instances can help improve performance, setting this value too high can strain your system resources, resulting in decreased performance.

Tuning Procedure

Finding the right settings for database pool size and runtime instances is a matter of experimenting with different settings to find the ones maximize available server resources without overloading resources and causing reduced performance.

> **Note:** You should optimize the flow pool size before tuning the database pool size. For information about optimizing the flow pool size, see **SettingDataflowPoolSize.dita#task_utx_h3t_tp**.

1. Begin by finding sample data to use as you test different settings. The sample dataset should be large enough that execution time is measurable and can be validated for consistency. The sample data should also be representative of the actual data you want to process. For example, if you are doing performance testing for geocoding, be sure that your test data has an equal number of records for all the countries you intend to geocode.
2. If you are testing a service or flow that requires the use of a database resource, such as postal databases or geocoding databases, make sure that you have the latest version of the database installed.
3. With sample data ready and the latest database resources installed, create a simple flow that reads data from a file, processes it with the stage you want to optimize, and writes to a file. For example, if you want to test performance settings for Validate Address, create a flow consisting of Read from File, Validate Address, and Write to File.
4. Set the database resource pool size to 1:
   a. Open Spectrum Management Console.
   b. Go to **Resources** > **Spectrum Databases**.
   c. Select the database resource you want to optimize and click the Modify button .
   d. In the **Pool size** field, specify 1.
   e. Click **OK**.
5. Set the stage's runtime instances to 1:
   a. Open the flow in Spectrum Enterprise Designer.
   b. Double-click the stage that you want to set to use multiple runtime instances.
   c. Click **Runtime**.

      > **Note:** Not all stages are capable of using multiple runtime instances. If there is no **Runtime** button at the bottom of the stage's window, the stage is not capable of using multiple runtime instances.

   d. Select **Local** and specify 1.
   e. Click **OK** to close the **Runtime Performance** window, then click **OK** to close the stage.

6. Calculate baseline performance by running the flow several times and recording the average values for:

   - Elapsed time
   - CPU utilization
   - Memory utilization

   Tip: You can use the Spectrum JMX console to monitor performance. For more information, see **Monitoring Performance with the Spectrum JMX Console** on page 231.

7. Run multiple instances of the job concurrently, if this is a use case that must be supported. Record elapsed time, CPU utilization, and memory utilization for each scenario.

   Tip: You can use a file monitor to run multiple instances of a job at once. For more information, see **Triggering a Flow with a Control File** on page 189.

8. Increment the database resource pool size and the stage runtime instances setting.
9. Restart the server.
10. Run the flow again, recording the elapsed time, CPU utilization, and memory utilization.
11. Continue to increment the database resource pool size and the stage runtime instances until you begin to see diminishing performance.
12. If you are testing geocoding performance, repeat this procedure using single country and multicountry input.

Setting Default Sort Performance Options

Sorting large data sets can be one of the most time-consuming operations performed during batch processing, so setting appropriate sort performance options can have a significant impact on the performance of your jobs. Sort performance options control memory and disk utilization, allowing you to take full advantage of the available memory and disk capacity.

There are two places where you can configure sort performance settings. The first is in Spectrum Management Console. This is where you specify default sort performance options for your system. The second place is in dataflow stages that perform a sort. The Sorter stage, Read from File, Write to File, and all other stages that include a sort operation, contain sort performance options. When you specify sort performance option in a stage, you override the default sort performance options, choosing different settings to apply to individual stages in a dataflow.

This procedure describes how to set the default sort performance options for jobs run on your Spectrum Technology Platform server.

1. Open Spectrum Management Console.
2. Go to **Flows** > **Defaults**.
3. Configure these options to control sort performance:

Option	Description
In memory record limit	Specifies the maximum number of data rows a sorter will hold in memory before it starts paging to disk. By default, a sort of 10,000 records or less will be done in memory and a sort of more than 10,000 records will be performed as a disk sort. The maximum limit is 100,000 records. Typically an in-memory sort is much faster than a disk sort, so this value should be set high enough so that most of the sorts will be in-memory sorts and only large sets will be written to disk. **Note:** Be careful in environments where there are jobs running concurrently because increasing the **In memory record limit** setting increases the likelihood of running out of memory.
Maximum number of temporary files	Specifies the maximum number of temporary files that may be used by a sort process. Using a larger number of temporary files can result in better performance. However, the optimal number is highly dependent on the configuration of the server running Spectrum Technology Platform. You should experiment with different settings, observing the effect on performance of using more or fewer temporary files. To calculate the approximate number of temporary files that may be needed, use this equation: `(NumberOfRecords × 2) ÷ InMemoryRecordLimit = NumberOfTempFiles`N **Note:** The maximum number of temporary files cannot be more than 1,000.
Enable compression	Specifies that temporary files are compressed when they are written to disk.

Note: The optimal sort performance settings depends on your server's hardware configuration. You can use this equation as a general guideline to produce good sort performance:

`(InMemoryRecordLimit × MaxNumberOfTempFiles ÷ 2) >= TotalNumberOfRecords`

Configuring Remote Component Options

A remote component is an underlying engine that performs a specific processing function, such as address validation, geocoding, or tax jurisdiction assignment. Some remote components can be configured to maximize performance. For example, a remote component might have options controlling how much reference data is cached in memory or how incoming data is matched to the reference data.

Each remote component is deployed into its own JVM. This means that the JVM configuration can be done through the remote component and independent of the server itself, allowing for flexibility of memory allocation and tuning of performance based on the characteristics of the remote component.

Remote component options affect all instances of the component as well as any stages that use that component. This is different from stage options, which can be modified at design time and at runtime.

Spectrum Universal Addressing component configuration

For U.S. address processing, there are several options controlling which reference data is cached in memory (for more information, see **Using Management Console to Create a Database Resource**).

- DpvMemoryModel: Controls which DPV files are in memory
- LacsLinkMemoryModel: Controls which LACS^{Link} files are in memory
- SuiteLinkMemoryModel: Controls which Suite^{Link} files are in memory

Spectrum Enterprise Geocoding component configuration

Spectrum Enterprise Geocoding has several options that can affect the performance of U.S. geocoding (for more information, see **Adding an Enterprise Geocoding U.S. Database Resource**).

- egm.us.multimatch.max.records: Specifies the maximum number of possible matches to return. A smaller number results in better performance, but at the expense of matches.
- egm.us.multimatch.max.processing: Specifies the number of searches to perform. A smaller number results in better performance, but at the expense of matches.
- FileMemoryLimit: Controls how much of the reference data is initially loaded into memory.

Heap Size Configuration for Elasticsearch

Elasticsearch is an underlying search technology used when performing data profiling in Discovery and when performing matching using the matching stages in Spectrum Enterprise Designer. Consider increasing the Elasticsearch heap size in these situations:

- In Discovery, you run multiple profiles concurrently or you run a profile having multiple tables

- You have flows that run multiple search index queries in parallel, each of which returns 1,000 or more candidates

To increase the Elasticsearch heap size, open this file in a text editor:

SpectrumDirectory\index\spectrum.vmargs

Increase the value in the `-Xmx` property. The default heap size is `-Xmx2048m`.

Optimizing Individual Stages

Optimizing Matching

Matching is typically one of the most time-consuming operations in any data quality implementation, making it important to ensure that matching is operating as efficiently as possible. There is always a balance between matching results and performance. If every record in a file is compared to every other record, you can be quite confident that all matches will be identified. However, this approach is unsustainable as the volume of data grows. For example, given an input file of 1 million records, matching each record to every other record would require nearly 1 trillion comparisons to evaluate each match rule.

Given that most records in a file do not match, the general approach to solving this problem is to define a match key and only compare those records that have the same match key. Proper match key definition is the most critical variable affecting performance of the matching engine. To define a proper match key, you must understand how the matching engine processes records and the options that are available.

The default matching method performs an exhaustive comparison of the record in a match queue to identify the maximum number of matches. Because of this, it is often the most time consuming way to do matching. Under the default matching method, the first record in the match queue becomes the suspect record. The next record is compared, and if it matches it is written out as a duplicate. If it does not match, it is added as a suspect, and the next record is compared to the two active suspects. Consider this match queue:

Unique ID	Match Key
1	123A
2	123A
3	123A
4	123A

Unique ID	Match Key
5	123A
6	123A
7	123A
8	123A
9	123A
10	123A

First, record 2 would be compared to record 1. Assuming it does not match, record 2 would be added as a suspect. Then record 3 would be compared to records 1 and 2, and so forth. If there are no matching records, the total number of comparisons would be 45. If some records match, the number of comparisons will be less. For a match queue of a given size N, the maximum number of comparisons will be N×(N-1)÷2. When the queue size is small this is not noticeable, but as the queue size grows the impact is significant. For example, a queue size of 100 could result in 4,450 comparisons, and a queue size of 500 could result in 124,750 comparisons.

Defining an Appropriate Match Key

To define an appropriate match key, consider these points:

- Most records do not match. Compare only records that are likely to match.
- Only records with the same match key will be compared.
- Performance is a key consideration:
  - The match key determines the size of the match queue.
  - For a given number of records, as the match queue size doubles, execution time doubles.
  - A "tight" match key results in better performance. A "tight" match key is one that is specific, containing more characters from possibly more fields.
  - A "loose" match key may result in more matches. A "loose" match key is one that is less specific, containing fewer characters from possibly fewer fields.

Finding a Balance Between Performance and Match Results

To find a good balance between performance and results, consider the match rule and the density of the data.

- Consider the match rules:

  - Fields requiring an exact match could be included in the match key.
  - Build an appropriate key for the match rule. For example, for a phonetic match rule, a phonetic match key is probably appropriate.
  - A match key will often consist of parts of all the fields being matched.
  - Be aware of the effects of missing data.
- Consider the density of the data:
  - For example, in address matching, the match key would likely be tighter if all the records are in a single town instead of a national dataset.
  - Consider the largest match queue, not just the average. Review the Match Summary report to find the largest match queue.
- When using transactional match, the same considerations apply to the SELECT statement in Candidate Finder.

Express Match Key

In a typical file, most of the duplicate records match either exactly or nearly exactly. Defining an express match key allows the matching engine to perform an initial comparison of the express match keys to determine that two records are duplicates. This can significantly improve performance by avoiding the need to evaluate all the field level match rules.

Intraflow Match Methods

The default Intraflow Match match method compares all records having the same match key. For a match queue size of N, the default method performs anywhere from N−1 to N×(N−1) comparisons. If all records match, the number of comparisons is N−1. If no records match the number of comparisons is N×(N−1). Usually the number of comparisons is somewhere in the upper part of this range.

If performance is a priority, consider using the sliding window match method instead of the default method. The sliding window match method compares each record to the next W records (where W is the window size). For a given file size N, the sliding window method performs no more than N×W comparisons. This can lead to better performance, but some matches may be missed.

Optimizing Candidate Finder

Candidate Finder selects candidate records from a database for comparison by Transactional Match. Since transactional match compares the suspect record to all of the candidate records returned by Candidate Finder, the performance of Transactional Match is proportional to the number of comparisons.

However, there are things you can do to improve the performance of Candidate Finder. To maximize the performance of Candidate Finder, a database administrator, or developer with extensive knowledge of the database schema and indexes, should tune the SQL SELECT statement in Candidate Finder. One of the most common performance problems is a query that contains a JOIN that requires a full

table scan. In this case, consider adding an index or using a UNION instead of a JOIN. As a general rule, SQL queries should be examined and optimized by qualified individuals.

Optimizing Transforms

The Transformer stage provides a set of predefined operations that can be performed on the input data. Generally, these predefined transforms run faster than custom transforms, since they are already compiled. However, when defining a large number of transforms, a custom transform will run faster. For example, to trim a number of fields, the custom transform below will typically run faster than nine separate trim transforms.

```
data['AddressLine1'] = (data['AddressLine1'] != null) ?
data['AddressLine1'].trim() : null;
data['AddressLine2'] = (data['AddressLine2'] != null) ?
data['AddressLine2'].trim() : null;
data['AddressLine3'] = (data['AddressLine3'] != null) ?
data['AddressLine3'].trim() : null;
data['AddressLine4'] = (data['AddressLine4'] != null) ?
data['AddressLine4'].trim() : null;
data['City'] = (data['City'] != null) ? data['City'].trim() : null;
data['StateProvince'] = (data['StateProvince'] != null) ?
data['StateProvince'].trim() : null;
data['PostalCode'] = (data['PostalCode'] != null) ?
data['PostalCode'].trim() : null;
data['LastName'] = (data['LastName'] != null) ? data['LastName'].trim()
 : null;
data['FirstName'] = (data['FirstName'] != null) ? data['FirstName'].trim()
 : null;
```

Optimizing Write to DB

By default the Write to DB stage commits after each row is inserted into the table. However, to improve performance enable the **Batch commit** option. When this option is enabled, a commit will be done after the specified number of records. Depending on the database this can significantly improve write performance.

When selecting a batch size, consider the following:

- **Data arrival rate to Write To DB stage:** If data is arriving at slower rate than the database can process then modifying batch size will not improve overall dataflow performance. For example, dataflows with address validation or geocoding may not benefit from an increased batch size.
- **Network traffic:** For slow networks, increasing batch size to a medium batch size (1,000 to 10,000) will result in better performance.
- **Database load and/or processing speed:** For databases with high processing power, increasing batch size will improve performance.
- **Multiple runtime instances:** If you use multiple runtime instances of the Write to DB stage, a large batch size will consume a lot of memory, so use a small or medium batch size (100 to 10,000).

- **Database roll backs:** Whenever a statement fails, the complete batch is rolled back. The larger the batch size, the longer it will take to perform the to rollback.

Optimizing Address Validation

Validate Address provides the best performance when the input records are sorted by postal code. This is because of the way the reference data is loaded in memory. Sorted input will sometimes perform several times faster than unsorted input. Since there will be some records that do not contain data in the postal code field, we recommend this sort order:

1. Country (Only needed when processing records for multiple countries)
2. PostalCode
3. StateProvince
4. City

Optimizing Geocoding

Geocoding stages provide the best performance when the input records are sorted by postal code. This is because of the way the reference data is loaded in memory. Sorted input will sometimes perform several times faster than unsorted input. Since there will be some records that do not contain data in the postal code field, the following sort order is recommended:

1. PostalCode
2. StateProvince
3. City

You can also optimize geocoding stages by experimenting with different match modes. The match mode controls how the geocoding stage determines if a geocoding result is a close match. Consider consider setting the match mode to the **Relaxed** setting and seeing if the results meet your requirements. The **Relaxed** mode will generally perform better than other match modes.

Optimizing Geocode US Address

The Geocode US Address stage has several options that affect performance. These options are in this file:

`SpectrumDirectory\server\modules\geostan\java.properties`

egm.us.multimatch.max.records	Specifies the maximum number of matches to return. A smaller number results in better performance, but at the expense of matches.
egm.us.multimatch.max.processing	Specifies the number of searches to perform. A smaller number results in better performance, but at the expense of matches.
FileMemoryLimit	Controls how much of the reference data is initially loaded into memory.

JVM Performance Tuning

There are JVM options that will yield performance gains with certain hardware. These are advanced options and can cause unexpected behavior when used incorrectly (possibly even JVM crashes). We recommend that you contact technical support if you want to explore increasing performance by tuning the JVM.

- On multiple CPU computers the `-XX:+UseParallelGC` option can be added to improve GC processing.
- We have also seen performance increases by adding these options, although on some hardware they have been known to cause JVM crashes.
  - `-Xmn512m`
  - `-XX:+AggressiveOpts`

Micro-batch Processing

Micro-batch processing is a technique where you include more than one record in a single service request. By including multiple records in a request instead of issuing separate requests for each record, you can significantly improve performance when processing a large collection of records through a service. Spectrum Technology Platform supports micro-batch processing for REST and SOAP web services.

Micro-batch end point

For AMER, use `amer-microbatch.precisely.com`. It has a 5 minute timeout allowing larger micro-batch sizes.

For APAC and EMEA, use the standard endpoint as they do not have a special micro-batch endpoint. This endpoint has a 30 second timeout, so the number of records in a micro-batch will need to be smaller.

Micro-Batch Size

For AMER using the special micro-batch endpoint, you may put as many records as can fit within a 5 minute timeout. You are charged for each record in the request even if it times out, so it is recommended to choose the number of records that can be processed in 4 minutes in case processing takes longer.

For APAC and EMEA using the standard endpoint, the maximum number of records allowed in a request depends on the service's category.

Using a Record ID

You may find it helpful to assign an ID to each record in a micro-batch so that you can correlate the records in the request with the records returned in the response. Use user fields to do this.

Monitoring Performance

Monitoring the performance of your dataflows enables you to tune performance by identifying performance bottlenecks. There are two ways you can monitor the performance: the Administration Utility and the Spectrum JMX Console.

The Administration Utility is a command line tool that provides access to many administrative functions, including a performance monitor. When enabled, the performance monitor writes performance data to a log file each time a dataflow is run and includes performance data for each stage in the dataflow.

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.

Monitoring Performance with the Administration Utility

The Administration Utility is a command line tool that provides access to many administrative functions, including a performance monitor. When enabled, the performance monitor writes performance data to a log file each time a dataflow is run and includes performance data for each stage in the dataflow.

1. Open the Administration Utility.
2. Type the following command:

   ```
   performancemonitor enabled set --e True --d DataflowName
   ```

   Where *DataflowName* is the name of the job or service you want to monitor.

Performance monitoring is now enabled for the dataflow you specified. When the dataflow runs, performance information will be written to the performance log.

The Performance Log

The performance log contains details about how long it takes for a job or service to run. It includes overall performance information for the job or service as well as performance information for each stage in the job or service dataflow. You can use this information to identify bottlenecks in your dataflow by looking at the execution time and processing time for each stage. A large difference between execution time and processing time means that the stage is spending time waiting for data

from upstream stages. This may indicate that an upstream stage is a bottleneck in the dataflow. Note that for sinks, a large difference between execution time and processing time does not necessarily indicate a performance problem because sinks typically have to wait for the first records from the rest of the dataflow.

To enable performance monitoring for a job or service, use the `performancemonitor enabled set` command in the Administration Utility.

The performance log is located on your Spectrum Technology Platform server in the following location:

SpectrumDirectory`\server\logs\performance.log`

The performance log contains one row for each run of a monitored job or service. It is a rolling log that consists of a maximum of five files. Each file is limited to 10 MB in size. Once this limit is reached, the oldest performance data is deleted when new performance data is logged.

> **Note:** The log file path name, maximum file size, and maximum number of files are specified by the PERFORMANCE `appender` settings in the `logback.xml` configuration file. For more information, see **Logback configuration file** on page 246.

Each entry in the performance log contains the following information. For ease of reading, line breaks and indentation are shown below. In the actual log, the entry is on one line.

```
Date Time [performance]
{
     "username":"UserName",
     "dataflowId":"DataflowName",
     "runMode":"BatchOrRealTime",
     "elapsedTime":Nanoseconds,
     "stageInfo":[
          {
               "stageName":"Name",
               "stageLabel":"Label",
               "options":{
                    OptionsList
               },
               "recordsRead":Count,
               "recordsWritten":Count,
               "executionTime":Nanoseconds,
               "processingTime":Nanoseconds
               "readBlockingTime":Nanoseconds
               "writeBlockingTime":Nanoseconds
               "readBlockingPercent":Percentage
               "writeBlockingPercent":Percentage
          }
     ]
}
```

Where:

username

The user who executed the job or service.

dataflowID

The name of the service or job as defined in Spectrum Enterprise Designer.

runMode

Indicates whether the log entry is for a job or a service. One of the following:

Batch	The log entry is for a job.
RealTime	The log entry is for a service.

elapsedTime

The time in nanoseconds that it took to run the job or service request.

stageInfo

Lists runtime information for each stage in the dataflow. The following information is listed for each stage:

stageName

The permanent name of the stage.

stageLabel

The user-defined name of the stage. The stage label is shown on the canvas in Spectrum Enterprise Designer.

options

If any options were specified at runtime, those options and their settings are listed here.

recordsRead

The total number of records that passed into the stage through all the stage's input ports.

recordsWritten

The total number of records that the stage wrote to all its output ports.

executiontime

The amount of time from when the stage processed its first record to when it processed its last record. This includes the time the stage was idle while waiting for data from other stages in the dataflow.

processingtime

The amount of time the stage spent actively processing records, not including the time it was idle while waiting for other stages in the dataflow.

readBlockingTime

The amount of time blocking while waiting to read the next record. A high read blocking time means that a prior process is taking longer than this stage and may indicate additional tuning is needed.

writeBlockingTime

The amount of time blocking while waiting to write the next record. A high write blocking time means that a prior process is taking longer than this stage and may indicate additional tuning is needed.

readBlockingPercent

The percentage of the total execution time that a stage was blocking on reading a record.

writeBlockingPercent
the percentage of the total execution time that a stage was blocking on writing a record.

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.

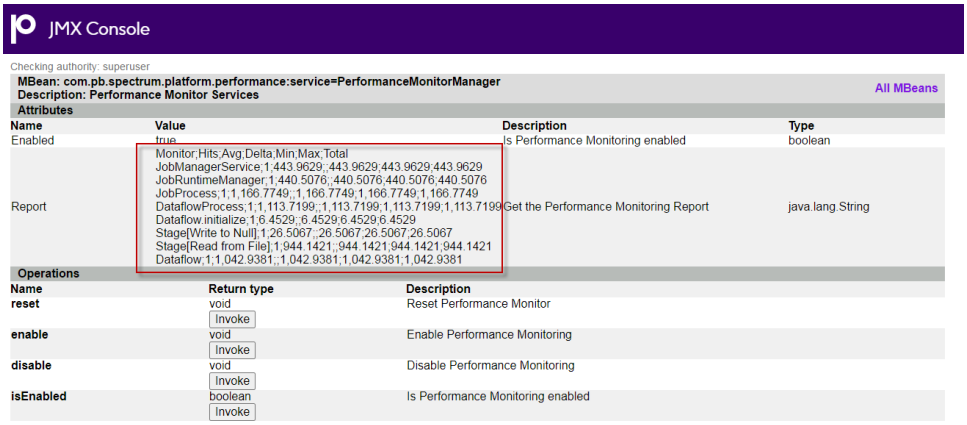
2. Log in using the admin account.
3. Under **Domain: com.pb.spectrum.platform.performance**, click **com.pb.spectrum.platform.performance:service=PerformanceMonitorManager**.
4. Click the **Invoke** button next to **enable**.
5. Click **Return to MBean View** to go back to the PerformanceMonitorManager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

JMX Performance Monitor Statistics

The Spectrum JMX Console's Performance Monitor Manager displays statistics about the performance of different parts of a dataflow execution, including the overall execution time, throughput, and execution time of individual stages. The statistics are reported in a semicolon-delimited format:



Tip: Put the data into a spreadsheet for easier viewing.

The first row is a header record consisting of the following columns:

Monitor	The item whose performance is being measured.
Hits	The number of times the item was run.
Avg	The average amount of time that the item spent processing a request, in milliseconds.
Delta	This statistic is not used.
Min	The shortest amount time that the item spent processing a request, in milliseconds.
Max	The longest amount of time that the item spent processing a request, in milliseconds.
Total	The total time spent processing, in milliseconds.

The most important items to review are the following.

Dataflow	The overall data flow response times and throughput.
ServiceRuntimeManager.borrow.*DataflowName*	Response time for handling concurrent requests, in milliseconds. This response time may be improved by modifying the dataflow pool size.
RemoteComponent.borrow.*RemoteComponentName*	Database resource performance directly impacted by the database resource pool size. This response time may go up as you increase your dataflow pool size. If this number increases to something significant such as ½ a millisecond increase the database resource pool size.
Stage[*StageName*]	The response times for each stage. This can help you identify bottleneck stages that are taking significantly longer to process than other stages. Bottleneck stages can potentially be addressed by

changing how the functionality in the existing stage is implemented and/or by increasing the runtime instances for that stage.

9 - Monitoring

In this section

Email Notification....235
Audit Log....238
System Log....240
Notification Log....243
Logging the Record that Caused a Flow to Fail....243
Transaction Limit Warnings....244
Viewing Version Information....245
Viewing server status....245
Viewing and Exporting License Information....246
Logback configuration file....246

Email Notification

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 on page 264

The Notifications feature enables you to have the system send a message to one or more email addresses when a designated number of exceptions are tied to a specific domain or metric.

Configuring license and expiration notification

This procedure describes how to specify when to send expiration notifications, and the recipients of the notification emails.

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. Click **Configure Notification**.
4. Check the **Send notification** box.
5. In the **Days before expiration** field, specify the number of days in advance that you want to be notified of a pending license, software, or data expiration. This is the default value. You can specify a different notification period for each license item on the **System** > **Licensing and Expiration** page.

   For example, if you want to be notified 30 days before items expire, specify `30`.

6. Under **Recipients**, click the Add button + and enter the email address you want to receive the expiration notification email. You can multiple email addresses if needed.
7. Click **Save**.

You have now specified recipients for the notifications and how far in advance of expiration to send the notification email. If you have not already done so, you must configure a mail server to use to send the emails. Notifications will not be sent until a mail server has been configured.

> **Note:** By default the system will send expiration notifications for all items that expire (licenses, databases, software components, for example). You can disable expiration notifications for specific items by going to **System** > **Licensing and Expiration**.

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.

Audit Log

Viewing the Audit Log

The audit log records the activities of users. It records events that occur when users create and modify objects on your system, as well as events that occur when users execute jobs or access services through the API or web services. Some examples of events recorded in the audit log include creating a dataflow, modifying a database connection, or running a job.

1. Open Spectrum Management Console.
2. Go to **System** > **Logs**.
3. Click **Audit Log**.

   The audit log lists the following information.

Column	Description
Severity	**Error** 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. **Warning** 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** Info events are typically seen during startup and initialization, providing information such as version information and which services were loaded.
Date/Time	The date and time of the event in the time zone of the Spectrum Technology Platform server.
User	The user account that performed the action.
Source	The software component that generated the event. This could be the name of a module or "Platform".

Column	Description
Event	The action that occurred. The platform events are listed below. In addition to these events, other events may appear in your audit log depending on the modules you have installed. **Create** — The object was created and saved to the server. **Create Version** — A new version of the dataflow was created in Spectrum Enterprise Designer. **Delete** — The object was removed from the server. **Delete Version** — The dataflow version was removed. Other versions may still exist. **Expose** — The dataflow was exposed, making it available for execution. **Item added** — The object was added to a folder on the server. **Item moved** — The object was moved to a different folder on the server. **Rename** — The object's name was modified and the object was saved. **Unexpose** — The dataflow was made unavailable for execution. It can still be edited in Spectrum Enterprise Designer. **Update** — The object was modified and saved.
Type	The part of the system that was modified by the event. Examples of types include the dataflow type (job, service, subflow, process flow) file servers, and access control settings. In some situations an object of the same name may appear multiple times with different values in the **Type** column. This is because one user action may generate multiple events in the system. For example, when you create a job in Spectrum Enterprise Designer, you will see a "Create" event for the job in the audit log, plus an "Item added" event for a FolderItem type that has the same name as the job. This indicates that the job was saved and placed in a folder on the system. The saving of the job and placement of the job into a folder are treated as two separate system events.
Object Name	The name of the item that generated the log entry. For example, the name of a dataflow. Object names may be user-defined, such as the name of a dataflow, or they may be defined by the system.

Audit Log Archiving

Audit log events are archived on a monthly basis to help prevent the audit log from growing too large. Every month, events that are six months old and older are moved to a compressed file in this folder:

SpectrumDirectory\server\repository\store\archive

You can move the compressed file to another location for permanent archiving if needed.

Monitoring system events through the Audit Log

Security administrators have two options for exporting and reviewing activity during specific timeframes. This tracking is easily integrated into logging.

auditlog export This command adds an activity log to all audit log files. The exported file can be in JSON or CSV format.

`auditlog export --s` *start_time* `--e` *end_time*

Times are in `yyyyMMddHHmmss` format. If no specific timeframe is specified, the default is the current day's start date and the time the command is issued.

auditlog info This command adds a count information file to the audit log files. The exported file is in JSON format.

`auditlog info --s` *start_time* `--e` *end_time*

Times are in `yyyyMMddHHmmss` format. If no specific timeframe is specified, the default is the current day's start date and the time the command is issued.

System Log

About the System Log

Messages in the System Log include information about server operations as well as requests made to services from the API and through web services.

The System Log records events on a Spectrum Technology Platform server. In a clustered environment, it records events on a node. The system log file is located on a Spectrum Technology Platform server or on any node in a clustered environment in the following location:

SpectrumDirectory\server\logs\spectrum-server.log

The system log contains rows for monitored events. By default, the system log file is a rolling log file with no maximum number of files. Each file is limited to 50 MB in size. Once this limit is reached in the `spectrum-server.log` file, the file is rolled over to `spectrum-server.log.1` and a new `spectrum-server.log` file becomes the active file. The oldest rollover file has the highest index number.

> **Note:** The log file path name and pattern, maximum file size, and maximum number of files are specified by the FILE `appender` settings in the `logback.xml` configuration file. For more information, see **Logback configuration file** on page 246.

To view the system log, see **Viewing System Events** on page 241.You can also view the system log by using a text editor to open the file on the server. In a clustered environment, you can use a text editor to open the file on a specific node.

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.
2. Go to **System** > **Logs**.
3. In the **System default logging level** field, 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 most intensive logging level can affect system performance. Therefore, you should select the least intensive setting that meets your particular logging requirements.

4. If you want to specify different logging levels for each service choose the logging level you want.

Notification Log

About the Notification Log

The Notification Log records notifications issued by the Spectrum Technology Platform server.

The notification log file is located on a Spectrum Technology Platform server in the following location:

SpectrumDirectory\server\logs\notification.log

The notification log contains a row for each notification. By default, the notification log file is a rolling log file with one rollover file. Each file is limited to 10 MB in size. Once this limit is reached in the `notification.log` file, the file is rolled over to `notification.log.1` and a new `notification.log` file becomes the active file.

> **Note:** The notification file path name and pattern, maximum file size, and maximum number of files are specified by the NOTIFICATION `appender` settings in the `logback.xml` configuration file. For more information, see **Logback configuration file** on page 246.

You can view the notification log by using a text editor to open the file on the server.

Logging the Record that Caused a Flow to Fail

When troubleshooting the cause of a flow failure it can be useful to examine the record that caused the failure. Flow failure records are written to a log file on the Spectrum Technology Platform server. The log file contains records that cause a stage within the flow to fail. The log does not capture records when the flow failure is due to other causes, such as malformed input records or expired licenses.

To enable the logging of records that cause flow failures:

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. Scroll down to this entry and click it:

   com.pb.spectrum.platform.configuration:manager=LoggingConfigurationManager

4. Set the attribute **LogLastRecordReadOnError** to **true** then click **set**.

Records that cause a flow to fail will now be logged in a new log file on the server:

SpectrumDirectory`/server/logserror_records.log`

> **Note:** Since this log may contain sensitive data, consider deleting the log file when you are done troubleshooting.

Transaction Limit Warnings

Transaction-based licenses place a limit on the number of transactions you can perform before you need to renew the license. When you have approximately 10% of your transaction limit remaining, warning messages begin appearing in the event log in Spectrum Management Console. For example, if you have a license that allows 1,000,000 transactions for the Validate Address service in Spectrum Universal Addressing, and you have performed 900,000 transactions, you will begin to see messages like this in the event log:

```
WARN  [ValidateAddress] license for feature(s): UNC/USA/RealTime has
100,000 transactions remaining
```

When you reach the limit, the feature is disabled and you will see messages like this in the event log:

```
ERROR [ValidateAddress] Usage limit exceeded for feature(s):
UNC/USA/RealTime
```

> **Note:** The system calculates the number of remaining transactions every few minutes and logs the warning message if necessary. If a job or a large number of transactions occurs and uses up the final 10% of remaining transactions at once, the remaining transactions may be used up before the system can display the warning message. In this situation, the warning message will not appear before the feature is disabled.

To view the number of transactions remaining on your license, open Spectrum Management Console, expand **System**, click **Licensing and Expiration**, then click the **License Info** tab.

To renew your license, contact your Precisely Account Executive.

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 server status

Use the URL-based server status check to quickly view overall status, such as memory use and CPU availability. This URL-based check provides a low-overhead and non-password protected way to check that your server is running and accessible.

1. In a web browser, specify one of these URLs:

   `https//server:port/dcg/status` or `http//server:port/dcg/status`

   Where:

   - *server* is the server name or IP address of your Spectrum Technology Platform server.
   - *port* is the HTTP or HTTPS port used by Spectrum Technology Platform.

     Note: By default, the server port is 8080 for HTTP and 8443 for HTTPS.

2. View the current status.

address	URL for the server you are viewing
cpuUsagePercentage	Portion of overall CPU allocation currently used, in decimal format
physicalMemoryUseagePercentage	Portion of memory currently used, in decimal format
heapMemoryUsagePercentage	Amount of runtime memory currently used, in decimal format
diskUsagePercentage	Amount of the overall disk space used, in decimal format

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.

Logback configuration file

Spectrum uses the Logback framework to provide server, notification, performance, and Hazelcast logging. The logback configuration file configures logging patterns and defines the maximum file size, file name and location, and maximum instances of rolling log files. The log configuration file also configures message patterns written to the console.

The logback configuration file defines `Logger` and `Appender` class instances used by Spectrum. In particular it defines the `file` path and name for each log file, the `pattern` for log entries, the `maxFileSize` for maximum log file size, and `rollingPolicy` and `maxIndex` for the rolling log files. For detailed information about logback architecture and configuration, you can refer to *The Logback Manual* linked to at the end of this topic.

The logback file is in the following location:

SpectrumDirectory`\server\conf\logback.xml`

Log files defined in this file include the following:

Log file	Appender name	Default path	For more information
Server	`FILE`	*SpectrumDirectory*`\server\logs\spectrum-server.log`	**About the System Log** on page 240

Log file	Appender name	Default path	For more information
Notification	NOTIFICATION	*SpectrumDirectory*\server\logs\notification.log	**About the Notification Log** on page 243
Performance	PERFORMANCE	*SpectrumDirectory*\server\logs\performance.log	**The Performance Log** on page 228
Hazelcast	HAZELCAST	*SpectrumDirectory*\server\logs\hazelcast.log	Distributed processing service

The logback file also defines `patterns` for console messages to `System.out` and `logger` instances. The logger instances specify logging levels in the production environment.

Related information

The Logback Manual

10 - Backup and Restore

In this section

About scheduled backups....249
Creating a Backup Manually....253
Restoring a Server....254
Restore a Spectrum repository backup....256

About scheduled backups

To back up your Spectrum Technology Platform server, you need to create a backup copy of the server's configuration database. The configuration database contains your security settings, dataflows, service options, data resource definitions, snapshots, and various configuration settings. If you were to lose your server due to a system failure or other disaster, you could use the backup of the configuration database to restore your configuration to another Spectrum Technology Platform server.

Important: Schedule backups to occur when there is little or no activity on the Spectrum Technology Platform server. While the backup is in progress, calls to services may time out and jobs may fail to run successfully.

Related reference

Backup properties on page 252
Below is a list of backup properties and their functions.

Scheduling system backups

This procedure describes how to configure Spectrum Technology Platform to create a backup on a regular schedule.

Learn more about the Spectrum system backup properties here: **Backup properties** on page 252

> **Note:** If you are running Spectrum Technology Platform in a clustered environment, you must shut down the entire cluster, configure all nodes identically, then restart the cluster.

1. Stop the Spectrum Technology Platform server.
2. Open this file in a text editor:

   SpectrumDirectory\server\conf\spectrum-container.properties

3. Specify these parameters:

   ```
   spectrum.backup.enabled=true
   spectrum.backup.cron=Interval
   ```

   Where:

   > **Note:** For more information on the cron configuration, visit **https://freeformatter.com/cron-expression-generator-quartz.html**.

   Interval

A cron expression that specifies how often to create the backup database. A cron expression consists of six space-separated values, with an optional seventh value:

Field	Valid Values	Valid Special Characters
Seconds	0-59	, - * /
Minutes	0-59	, - * /
Hours	0-23	, - * /
Day of the month	1-31	, - * ? / L W
Month	1-12 or JAN-DEC	, - * /
Day of the week	1-7 or SUN-SAT	, - * ? / L #
Year (Optional)	1970 - 2099	, - * /

For example, this expression would back up the configuration database every day at 10:00 AM:

```
spectrum.backup.cron=0 0 10 * * ?
```

This expression would back up the configuration database on the first day of the month at 2 AM:

```
spectrum.backup.cron=0 0 2 1 * ?
```

The special characters are:

*****

Specifies all values. For example, if you use `*` in the day-of-the-month field, it means every day of the month.

?

Specifies no specific value. This is used in combination with other fields. For example, if you want to run a backup on the first day of the month and don't care which day of the week the first is on, you would specify `?` in the day-of-the-week field and `1` in the day-of-the-month field.

-

Specifies a range of values. For example, `SAT-SUN` means the Saturday through Sunday.

,

Separates multiple values. For example, `15,30` in the day-of-the-month field means the 15th day of the month and the 30th day of the month.

/

Specifies increments. For example, `0/3` in the hour field means the backup will occur at midnight then every three hours.

L

Specifies "last", which has different meaning depending on the field in which it is used. When used in the day-of-the-month field, it means the last day of the month. When used alone in the day-of-the-week field, it means Saturday. However, when used in the day-of-the-week field in combination with a day, it means the last day-of-the-week in the month. For example, `6L` means the last Friday of the month.

W

Use this value in the day-of-the-month field to specify the weekday nearest to a given day. For example, `15W` means the nearest weekday to the 15th day of the month.

Destination

The directory where you want to save the backup database. For example,

```
spectrum.backup.repository.directory\\\\exampleserver1\\Shared\\Backup
```

You must use the escape character \ when specifying a backslash in the path.

> **Note:** If you are using Spectrum Technology Platform in a clustered environment, you should specify a centralized location as the backup destination. This is because in a clustered environment, scheduled backups occur on a random node in the cluster. By specifying a centralized location it will be easier to retrieve the latest backup from the cluster.

4. To backup your Neo4j repository, specify these properties:

```
spectrum.backup.repository.enabled=true
spectrum.backup.repository.databaseURL=URLOrHostMachine
spectrum.backup.repository.directory=Destination
```

5. To backup your Elasticsearch index repository, specify these properties:

```
spectrum.backup.index.enabled=true
spectrum.backup.index.directory=Destination
```

6. Save and close the properties file.
7. Start the Spectrum Technology Platform server.
8. Optional: If you are using Spectrum Technology Platform in a clustered environment, repeat this procedure for each node in the cluster.

Note: You must specify identical values for *all* properties on *all* cluster nodes.

9. Some modules store additional data that is not backed up as part of the Spectrum Technology Platform scheduled backup process. You must back up this data manually, or create a separate process to back up this data.
10. Back up module-specific data for any of these modules if you have them installed.

Module	Items to Back Up
Advanced Matching, Data Normalization, and Universal Name	Back up the contents of these subfolders located in *SpectrumDirectory*\server\modules: • cdqdb • lucene • matcher • parser • searchindex • tables
Context Graph	Backup and restore for Context Graph models is performed in the Neo4j Admin tool. For more information **Neo4j Admin Manual**.
Spatial	Back up your named resources, data, and configuration files.

Backup properties

Below is a list of backup properties and their functions.

General backup properties

Property	Description
spectrum.backup.enabled	Enable or disable all system backups
spectrum.backup.cron	Quartz cron configuration for scheduled backups: For more information on the cron configuration, visit **https://freeformatter.com/cron-expression-generator-quartz.html**.

Neo4j repository backup properties

Property	Description
`spectrum.backup.repository.enabled`	Enable or disable the backup of the Neo4j repository, specifically: Overrides the general enabled flag (`spectrum.backup.enabled`)
`spectrum.backup.repository.databaseURL`	URL/Host of the machine where Neo4j repository runs; Do not modify unless Neo4j is running on a different machine than the server
`spectrum.backup.repository.directory`	Directory where Neo4j backup files are stored. By default, this location is `../server/backup/repository`. This location has changed from the previous default location, which was `../server/app/repository/store/backup` in previous releases. In clustered setups, we suggest that you point this directory to a network share location.

Elasticsearch index backup properties

Property	Description
`spectrum.backup.index.enabled`	Enable or disable the backup of the Elasticsearch index repository, specifically: Overrides the general enabled flag (`spectrum.backup.enabled`)
`spectrum.backup.index.directory`	Directory where backup files are stored: By default, this location is `../server/backup/index`. IMPORTANT: In a clustered environment, this property **must** point to a network share location.

To restore an Elasticsearch snapshot, invoke the **restoreSnapshot** operation using the Index Snapshot/Backup Manager in the JMX Console:

```
MBean:  com.pb.spectrum.platform.common:manager=IndexSnapshotManager
```

Creating a Backup Manually

To back up your Spectrum Technology Platform server, you need to create a backup copy of the server's configuration database. The configuration database contains your security settings, dataflows,

service options, data resource definitions, snapshots, and various configuration settings. If you were to lose your server due to a system failure or other disaster, you could use the backup of the configuration database to restore your configuration to another Spectrum Technology Platform server.

To manually create a of the Spectrum Technology Platform configuration database, use the Administration Utility's `server backup` command. For more information, see **server backup** on page 503.

In addition, some modules have data that is not included in the Administration Utility backup process. You must back up this data separately:

Module	Items to Back Up
Advanced Matching, Data Normalization, and Universal Name	Back up the contents of these subfolders located in *SpectrumDirectory*\server\modules: • cdqdb • lucene • matcher • parser • searchindex • tables
Context Graph	Backup and restore for Context Graph models is performed in the Neo4j Admin tool. For more information **Neo4j Admin Manual**.
Spatial	Back up your named resources, data, and configuration files.

Restoring a Server

If you lose your server due to a severe system failure or other disaster, you can restore your server using a backup of the configuration database. In order to have a backup you must have either created a backup manually or have configured Spectrum Technology Platform to create backups on a regular schedule. By default, Spectrum Technology Platform does not create backups of the configuration database.

> **Note:** This procedure is intended to be used in situations where you have a single Spectrum Technology Platform server. If you have a cluster of Spectrum Technology Platform servers and you need to restore a single node, install a new server and add it to the node. The configuration of the cluster will automatically be applied to the new node, in effect restoring

the node. The only scenario where you would need to restore from a backup in a clustered environment would be in the event of a complete loss of all nodes in the cluster.

1. Install a new Spectrum Technology Platform server. For more information, see the *Installation Guide*.
2. If the server is running, stop the server.
3. Obtain the backup zip file and unzip it to this location, overwriting existing files:

   - *SpectrumFolder*\repository\data\databases

   This will replace the existing `graphdb` folder.

4. Restore the module-specific data for any modules you have installed.

Module	Items to Back Up
Advanced Matching, Data Normalization, and Universal Name	Restore the contents of these subfolders located in *SpectrumFolder*\server\modules: • cdqdb • lucene • matcher • parser • searchindex • tables
Context Graph	Restore your models. In addition to restoring your models, restore these two property files: • *SpectrumFolder*\server\modules\hub\hub.properties • *SpectrumFolder*\server\modules\hub\db\neo4j.properties
Spatial	Restore your named resources, data, and configuration files.

5. Start the server.
6. Wait for the server to fully start.
7. Stop the server.
8. Apply all the updates for the platform and any modules you have installed.
   For a listing of updates, see the **Update Summary** on the Precisely support website.

Restore a Spectrum repository backup

Use this procedure to restore the Spectrum respository from a backup created by the Admiinistration Uitility `server backup` CLI command.

1. Stop the Spectrum Technology Platform server.
2. Unzip the `backup.zip` file obtained from running the CLI `server backup` command.
3. Open the Command Prompt as administrator, and on the command line navigate to the *`SpectrumDirectory`*`/repository/bin` directory.

   On the Linux platform, use the `ls -l` command to check that `neo4j-admin` is executable.
4. Run the following command:

Platform	Command
Windows	`neo4j-admin.bat restore --from=`*`unzip-path`* `--database=graphdb`
Linux	`./neo4j-admin restore --from=`*`unzip-path`* `--database=graphdb --force`

   In the command line, *`unzip-path`* is the directory in which you expanded the contents of the `backup.zip` file in step **2** on page 256.
5. After the `restore` command completes, verify that a folder named `graphdb` is present in `databases` folder in the following path:

   `SpectrumDirectory``/repository/data/databases/graphdb`
6. Start the Spectrum Technology Platform server.

Related reference

server backup command on page 503

11 - Settings

In this section

Data Stewardship Settings.....258
Context Graph Settings.....268

Data Stewardship Settings

Introduction

Data Stewardship Settings provides the following tools for users with write permissions:

Tool	Description
Lookups	The Lookups tool provides a way for you to select from a list of values for a specific field when updating records in the Data Stewardship Portal **Editor** page.
Domains	Domains specify the kind of data being evaluated.
Metrics	Metrics specify the way in which data is measured.
Notifications	The Notifications feature enables you to have the system send a message to one or more email addresses when a designated number of exceptions are tied to a specific domain or metric.
Data Quality Reporting	The Data Quality Reporting settings configure preferences for tracking pass/fail conditions and KPIs.
Search Tools Services	Search Tools Services includes all services for which you are licensed, such as ValidateAddress, GetPostalCodes, and so on. You can use these search tool services in the Data Stewardship Portal **Editor** to look up and validate exception data that you are attempting to correct or augment.
Options	These settings provide preferences for audit logs and progress tracking.
Approval Flow Types	The **Approval Flow Types** setting allows you to define types that associate records to an approval flow.

Accessing Data Stewardship Settings

Complete this procedure to access to access the **Data Stewardship Settings** page.

The **Data Stewardship Settings** page is located in the Management Console application.

1. On the Spectrum Technology Platform **Home** page, click **Platform Client Tools** > **Web** > > **Open Management Console**.

   Alternatively, navigate in your web browser to:

   `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 bySpectrum Technology Platform. By default, the HTTP port is 8080.

2. Enter a valid user name and password.
3. Click **Resources** > **Data Stewardship Settings**.

Lookups

The Lookups tool provides a way for you to select from a list of values for a specific field when updating records in the Data Stewardship Portal **Editor** page.

This feature is particularly useful when you have several records with data in the same field that you want to change. For example, you could have a set of exception records that contain banking data. One of the fields in that data could consist of codes that represent what kind of account is tied to the record (1=checking, 2=savings, 3=money market, and so on). whose addresses include ISO codes instead of names in the Country field, making those addresses unable to be validated. To correct this, you could create a lookup that provides ISO codes with their respective country names and make the corrections in the Data Stewardship Portal **Editor**, where you select the ISO code from a list that then populates the field with the country name tied to that ISO code.

Another benefit of using this tool is that it limits the options available for corrections, which reduces the possibility of further error. Using the same example of country names being incorrect or missing, by creating a lookup that provides a list of country names instead of requiring those names to be manually entered for each exception record, you ensure that those names are spelled correctly and are more likely to be validated when they are reprocessed.

What are lookups?

A lookup is made up of values or value/label pairs that contain data to replace existing, problematic data in a dataflow that is producing exception records.

The value in a lookup will replace the problematic data. The label in a lookup is displayed in the list that you select from when using the lookup table to correct records on the Data Stewardship Portal **Editor** page.

What is the Lookup Process?

The lookup process involves three steps after you have reviewed exceptions and identified a recurring issue among those exceptions (such as invalid data in a country field):

- Create the lookup using values and/or labels of accurate data that will overwrite the bad data.
- Using the Write Exceptions stage in the dataflow that is producing the exception records, point the problematic field to the lookup you created and rerun the dataflow.
- Correct the exception records on the Data Stewardship Portal **Editor** page by overwriting bad data in the problematic field with good data from the lookup.

Creating Lookups

You can populate a lookup by manually entering single value/label pairs or by copying value/label pairs from an external source and pasting it into the **Add many** dialog box. The external source can be a spreadsheet, a text file, or virtually any other file as long as the information is presented in one or two columns with either a comma, tab, or semicolon delimiter.

Note: If you only include values in your lookup table, the values will also be used as labels.

1. On the **Management Console** page, click **Resources** > **Data Stewardship Settings**.
2. On the sidebar, click **Lookups**.
3. Click the Add lookup button .
4. Enter a name for the new lookup in the text box.
5. Add a value/label pair.

   To manually add single value/label pairs:

   a) Click the Add lookup value button .
   b) Enter **Value** and/or **Label** for the lookup pair.

   To add lists of value/label pairs:

   a) Click the Add many button to open the **Add many** dialog box.
   b) Configure **First column**, **Separator**, and **Second column** according to your list.

      If you are pasting data from Microsoft Excel, use the Tab separator. If you select the wrong separator, the tool will import the entire line including the separator as the value or label (according to the **First column** selection).

c) Type or paste in rows of values, separators, and labels.

If a row has no entry in the first column, the separator must still precede the second column entry.

After all value/label pairs have been added you can sort them in ascending or descending order on either the **Value** or the **Label** column.

Note: Once you have sorted the list, you can only change the order. You cannot return to the original in which values were initially added.

6. Repeat step **5** on page 260 to add additional value/label pairs.
7. Click **Save**.

   If you used the **Add many** function, any previously existing values or value/label pairs for that lookup will be deleted. However, you can afterwards manually add additional values or value/label pairs.

Assigning Lookups

After creating a lookup, you need to assign that lookup to the field with problematic data in the Write Exceptions stage of the dataflow.

1. In Enterprise Designer, open the dataflow that is producing the exception records.
2. Open the Write Exceptions stage.
3. In the **Lookup name** column for the field with problematic data, select the lookup that contains the new, accurate data from the drop-down list.
4. Click **OK**.
5. Save and rerun the dataflow.

Correcting Records

After creating a lookup and assigning that lookup to a field in the dataflow, you need to correct the exception records in Data Stewardship.

1. On the Data Stewardship **Editor** page, select the dataflow that is producing the exception records.
2. For the first problematic record, click the field that you assigned the lookup to.
3. Click the drop-down button in that field and select the correct label for that record.

   Remember: This label is not necessarily the same as the value. For example, if you want your field to have a value of "California" you might click a label that says "CA".

4. Repeat step **3** on page 261 for each problematic record.
5. Save the changed exceptions.

Modifying or Deleting Lookups

1. On the Management Console **Resources** > **Data Stewardship Settings** page, click **Lookups** in the sidebar.
2. Check the box next to the appropriate lookup.
3. Modify or delete the lookup.

Option	Description
To modify the lookup	Click the Edit lookup button , modify lookup pairs as necessary, and click **Save**.
To delete the lookup	Click the Delete lookup button .

4. Click the **Save** button.

Domains

Domains specify the kind of data being evaluated. This is used for reporting purposes to show which types of exceptions occur in your data. For example, if the condition evaluates the success or failure of address validation, the data domain could be "Address". If the condition evaluates the success or failure of a geocoding operation, the data domain could be "Spatial".

> **Note:** The domains you establish here will serve as default options both for Data Stewardship Settings and the Exception Monitor stage.

You can select one of the predefined domains or you can define your own domain.

- To define your own domain, click the Add item button and configure options as necessary.
- To edit an existing domain, select it, click the Edit item button , and make desired changes.
- To delete a domain, select it and click the Delete item button .
- To filter the list of domains, enter search data in the **Filter** field. The table updates dynamically.

The following predefined domains are initially available. These can be edited as required.

- Account—The condition checks a business or organization name associated with a sales account.
- Address—The condition checks address data, such as a complete mailing address or a postal code.
- Asset—The condition checks data about the property of a company, such as physical property, real estate, human resources, or other assets.
- Date—The condition checks date data.
- Email—The condition checks email data.

- Financial—The condition checks data related to currency, securities, and so forth.
- Name—The condition checks personal name data, such as a first name or last name.
- Phone—The condition checks phone number data.
- Product—The condition checks data about materials, parts, merchandise, and so forth.
- Spatial—The condition checks point, polygon, or line data which represents a defined geographic feature, such as flood plains, coastal lines, houses, sales territories, and so forth.
- SSN—The condition checks U.S. Social Security Number data.
- Uncategorized—Choose this option if you do not want to categorize this condition.

Metrics

Metrics specify the way in which data is measured. This is used for reporting purposes to show which types of exceptions occur in your data. For example, if the condition is designed to evaluate the record's completeness (meaning, for example, that all addresses contain postal codes) then you could specify "Completeness" as the data quality metric.

> **Note:** The metrics you establish here will serve as default options both for Data Stewardship Settings and the Exception Monitor stage.

You can select one of the predefined metrics or you can define your own metric.

- To define your own metric, click the Add item button and configure options as necessary.
- To edit an existing metric, select it, click the Edit item button, and make desired changes.
- To delete a metric, select it and click the Delete item button.
- To filter the list of metrics, enter search data in the **Filter** field. The table updates dynamically.

The following predefined metrics are initially available. These can be edited as required.

- Accuracy—The condition measures whether the data could be verified against a trusted source. For example, if an address could not be verified using data from the postal authority, it could be considered to be an exception because it is not accurate.
- Completeness—The condition measures whether data is missing essential attributes. For example, an address that is missing the postal code, or an account that is missing a contact name.
- Consistency—The condition measures whether the data is consistent between multiple systems. For example if your customer data system uses gender codes of M and F, but the data you are processing has gender codes of 0 and 1, the data could be considered to have consistency problems.
- Interpretability—The condition measures whether data is correctly parsed into a data structure that can be interpreted by another system. For example, social security numbers should contain only numeric data. If the data contains letters, such as `xxx-xx-xxxx`, the data could be considered to have interpretability problems.
- Recency—The condition measures whether the data is up to date. For example, if an individual moves but the address you have in your system contains the person's old address, the data could be considered to have a recency problem.

- Uncategorized—Choose this option if you do not want to categorize this condition.
- Uniqueness—The condition measures whether there is duplicate data. If the dataflow could not consolidate duplicate data, the records could be considered to be an exception.

Notifications

The Notifications feature enables you to have the system send a message to one or more email addresses when a designated number of exceptions are tied to a specific domain or metric.

A notification email includes a link to the failed records in the Exception Editor of the Data Stewardship Portal, where users can manually enter the correct data. To stop sending notifications to a particular email address, remove that address from the list of recipients in the Send notification to line of the Edit domain page.

> **Note:** A mail server must be configured before you can successfully use a notification from within Data Stewardship Settings.

Related tasks

Configure Notifications on page 264
This procedures steps through configuring notifications for domains or metrics.

Configuring a Mail Server on page 235

Configure Notifications

This procedures steps through configuring notifications for domains or metrics.

A mail server must be configured before you can successfully use a notification from within Data Stewardship Settings.

1. In **Management Console**, click **Resources** > **Data Stewardship Settings**.
2. Click either **Domains** or **Metrics**.
3. Select the check box next a domain or metric, and click the Edit item button .
4. In the **Send notification to** box, select user names from the drop-down list or enter new email addresses to which notifications should be sent.

   Users are configured in Management Console.
5. In the **Number of exceptions to trigger notification box**, select the number of exception records that should trigger a notification.
6. In the **Subject** box, enter the text that should be sent as the subject of the notification.
7. In the **Message** box, enter the message that should appear in the body of the notification.

   You can use the following variables in the message to relay important information about the exceptions:

   - `${jobID}`—The ID number of the job that produced the exception records.
   - `${jobName}`—The name of the job that generated the exception records.

- `${userName}`—The name of the user whose job that generated the exception records.
- `${stageLabel}`—The name of the dataflow stage that produced the exception records.
- `${link}`—A link to the Editor page in the Data Stewardship Portal, showing records for a particular dataflow.

8. Check the **Send reminder** check box if you want to send a reminder notification, and select the number of days that should pass before the reminder is sent.
9. In the **Send reminder to** box, select user names or enter an email addresses.
10. In **Reminder Subject**, enter the text that should be sent as the subject of the reminder notification.
11. In **Reminder Message**, enter the message that should appear in the body of the reminder notification.

    The reminder uses an additional variable:

    - `${Count}`—The number of exceptions for the specified dataflow or stage that have yet to be resolved.

12. Check the **Remind daily** check box if you want a reminder notification to be sent every day until the exceptions are resolved.

Related concepts

Notifications on page 264
The Notifications feature enables you to have the system send a message to one or more email addresses when a designated number of exceptions are tied to a specific domain or metric.

Related tasks

Configuring a Mail Server on page 235

Data Quality Reporting

The Data Quality Reporting settings configure preferences for tracking pass/fail conditions and KPIs.

1. On the **Management Console** page, click **Resources** > **Data Stewardship Settings**.
2. On the sidebar, click **Data Quality Reporting**.
3. Check the **Data quality reporting** check box to track pass or fail conditions in the Exception Monitor stage.

   If you turn off this option, the **Data Quality** page Data Stewardship will contain no data. Likewise, the "Report only" field in the Exception Monitor stage for all dataflows will be disabled.
4. In the **Retention** drop-down, select how long, in months, that data should be retained.
5. Configure Key Performance indicators.

   For more information, see **Configuring Key Performance Indicators** on page 266.

Configuring Key Performance Indicators

The **KPI Configuration** section of the Data Quality Reporting tab enables you to designate key performance indicators (KPIs) for your data and assign notifications for when those KPIs meet certain conditions.

1. On the **Management Console** page, click **Resources** > **Data Stewardship Settings**.
2. On the sidebar, click **Data Quality Reporting**.
3. Click the Add a KPI button .
4. Enter a **Name** for the key performance indicator.

   This name must be unique on your Spectrum Technology Platform server.
5. Select one of the data quality **Metrics** for the key performance indicator.

   If you do not make a selection, this key performance indicator will be tied to all **metrics**.
6. Select a **Dataflow name** for the key performance indicator.

   If you do not make a selection, this key performance indicator will be tied to all Data Stewardship dataflows.
7. Select a **Stage label** for the key performance indicator.

   If you do not make a selection, this key performance indicator will be tied to all Data Stewardship stages in your dataflows.
8. Select a data **Domain** for the key performance indicator.

   If you do not make a selection, this key performance indicator will be tied to all **domains**.

   Note: Selecting a domain disables the **Condition** option.
9. Select a **Condition** for the key performance indicator.

   If you do not make a selection, this key performance indicator defaults to All.

   Note: Selecting a condition disables the **Domain** option.
10. Select a **KPI period** to designate the intervals for which you want Data Stewardship to monitor your data and send notifications.

    For example, if you enter "1" select Month, a KPI notification will be sent when the percentage of exceptions has increased per the threshold or variance over a month-to-month interval.
11. Provide a percentage for either a **Variance** or a **Threshold**.

    Variance values represent the increased percentage of failures in exception records since the last time period. Threshold values represent the percentage of failures at which you want the notifications to be sent. Its value must be 1 or greater.
12. Select email addresses from the list or enter email addresses for the **Recipients** who should be notified when these conditions are met.

When possible, this field auto-completes as you enter email addresses. You do not need to separate addresses with commas, semicolons, or any other punctuation.

13. Enter the **Subject** you want the notification email to use.
14. Enter the **Message** you want the notification to relay when these conditions are met.
15. Click **OK**.
    The new KPI is added to the list of KPIs. You can sort KPIs on any of the columns containing data.
16. Click **Save**.

You can modify or remove KPIs by selecting a KPI and clicking the Edit selected KPI button or the Delete selected KPI button .

Search Tools Services

Search Tools Services includes all services for which you are licensed, such as ValidateAddress, GetPostalCodes, and so on. You can use these search tool services in the Data Stewardship Portal **Editor** to look up and validate exception data that you are attempting to correct or augment.

1. On the **Management Console**, click **Resources** > **Data Stewardship Settings**.
2. On the sidebar, click **Search Tools Services**.
   The list of available services is based on user permissions and populated by modules and services that are licensed within Spectrum Technology Platform.
3. Optional: Enter text in the **Filter** box to narrow the list of services based on filter criteria.
4. Check the check box next to a service to make it available in the the Data Stewardship Portal **Editor**.
5. Check the check box in the **Premium** column to indicate to users that they will accrue additional fees when they use a service (such as Dun & Bradstreet services).

Options

These settings provide preferences for audit logs and progress tracking.

1. On the **Management Console**, click **Resources** > **Data Stewardship Settings**.
2. On the sidebar, click **Options**.
3. Configure options on this page.

Select	To
Audit exception events	Have Data Stewardship maintain a log of when exception records are created, read, updated, or deleted.

Select	To
Track progress	Track when exception records are approved in Data Stewardship. If you turn this option off, progress charts will not appear on the Data Stewardship.

Approval Flow Types

The **Approval Flow Types** setting allows you to define types that associate records to an approval flow.

Approval flow types are used when defining conditions in the Exception Monitor stage. If a record meets the criteria of a condition, the record will then be associated to that approval flow. For more information, see **Approval Flows** on page 15.

- To create a new approval flow type, click the Add item button +.
- To delete existing approval flow types, check the check box next to type names that you want to delete, and click the Delete item button .

Context Graph Settings

Context Graph and Context Graph Repository settings are accessed from the **Resources** menu on the **Management Console** page. These settings enable users with Administration view and modify permissions to set preferences for audit logs and query timeout as well as to configure connections to the Neo4j server used by Context Graph.

Accessing Context Graph Settings

Complete this procedure to access to access the **Context Graph Settings** page.

The **Context Graph Settings** page is located in the Management Console application.

1. In a web browser, navigate 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 bySpectrum Technology Platform. By default, the HTTP port is 8080.

2. Enter a valid user name and password.
3. Click **Resources** > **Context Graph Settings**.
4. Click the **Save** button after you make any changes to the settings.

Related reference

Context Graph Settings on page 269
These options configure audit log settings and query timeout for Context Graph.

How to access Context Graph Repository Options

Complete this procedure to access to access the **Context Graph Repository Options** page.

The **Context Graph Repository Options** page is located in the Management Console application.

1. In a web browser, navigate 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 bySpectrum Technology Platform. By default, the HTTP port is 8080.

2. Enter a valid user name and password.
3. Click **Resources** > **Context Graph Repository Configuration**.

Related reference

Context Graph Repository Configuration options on page 270
These options define connection settings for the Neo4j server used by Context Graph.

Context Graph Settings

These options configure audit log settings and query timeout for Context Graph.

To access these Management Console settings, click **Resources** > **Context Graph Settings**.

Audit metadata events	Check this check box to log when metadata is created, updated, or deleted.
Audit model events	Check this check box to log when models are created, updated, or deleted.
Include read events	Check this check box to log when models are viewed. Note that including this data can significantly impact storage limitations for the audit log. Audit model events must be enabled to include this activity.
Query timeout	Check this check box to specify in seconds how long the server should wait for a query to complete. Allowed values are between 1 and 100000.

Related tasks

Accessing Context Graph Settings on page 268
Complete this procedure to access to access the **Context Graph Settings** page.

Context Graph Repository Configuration options

These options define connection settings for the Neo4j server used by Context Graph.

Authentication type	• None—Choose this option if credentials are not required to access Neo4j. • Basic—Choose this option to specify credentials to access Neo4j.
User name	Specifies a user name used to read and write data in the Neo4j database. This option is not visible when **Authentication type** is set to None.
Password	Specifies the password that corresponds to the **User name**. This option is not visible when **Authentication type** is set to None.
Secured connection	Select this check box to use HTTPS to connect to the Neo4j server.
IP Address\Hostname Port	This table specifies the IP addresses the Neo4j server. It also specifies the advertised port address for the `Bolt connector` on the Neo4j server. The default port setting used by Context Graph for the Bolt connector address is `7700`. For more information, refer to the *Spectrum Technology Platform 2022.1.0 Installation Guide*. Click the Add address Edit address or Delete address buttons to add, edit, or delete addresses shown on the list. For a clustered configuration, add the Neo4j Server address for each node and the port number specified by the advertised address for the Bolt connector on a node.
Test	Click this button to test the connection to the Neo4j database server. The Neo4j server must be running to perform this test.

Related tasks

How to access Context Graph Repository Options on page 269
Complete this procedure to access to access the **Context Graph Repository Options** page.

12 - Administration Utility

In this section

Getting Started with the Administration Utility....272
Audit Log Information....276
Spectrum Business Glossary....278
Data Stewardship....280
Context Graph....281
Data Sources....304
Dataflows....318
Entities....326
Folders....327
Information Extraction....329
Jobs....339
Spectrum Lineage & Impact Analysis....349
Spectrum Machine Learning....350
Match Keys....352
Match Rules....355
Best of Breed Rules....358
Metadata Connections....361
Notification....363
Open Parser Cultures....367
Open Parser Domains....368
Performance Monitor....370
Permissions....373
Physical and Logical Models....374
Process Flows....385
Product Data....394
Profiles....399
Roles....404
Scorecard....409
Search Indexes....413
Services....422
Spectrum Databases....426
Service pool size....497
System....499
Tables....508
Tokens....511
User Accounts....513

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.

Install the Administration Utility

Download and install the Administration Utility from the Spectrum Technology Platform **Home** page. This topic also describes how to connect to the Spectrum Technology Platform server.

> **Note:** The Administration Utility requires Java 11 or later. Verify that Java 11 is in the system PATH variable before running the Administration Utility.

If you are reinstalling an existing installation of the Administration Utility, first back up the `cli.properties` file if it exists and any other CLI configurations. These are located in the same folder in which the CLI files are located. Add the backed up file and other configurations to the updated CLI after you reinstall the Administration Utility.

1. On the Spectrum Technology Platform **Home** page, click **Platform Client Tools**.
2. Click **Command Line**.
3. Under **Administration Utility**, click **Download** and download the zip file to the computer where you want to use the Administration Utility.
4. Extract the contents of the zip file.
5. To launch the command line interface, do one of the following:
   - If you are running the server on a Linux system, execute `cli.sh`.
   - If you are running the server on a Windows system, execute `cli.cmd`.

   > **Note:** If necessary, modify the `.sh` or `.cmd` file to use the path to your Java installation.

6. Connect to the Spectrum Technology Platform server by typing this command:

   `connect` --h *servername*:*port* --u *username* --p *password* --s *SSLTrueOrFalse*

   For example,

   `connect` --h myserver:8080 --u admin --p myPassword1 --s true

7. Once you are connected you can run commands. Some tips:
   - For a list of available commands, type `help` or press the tab key.

- To auto-complete a command, type the first few characters then press the tab key. For example, typing `us` then pressing the tab key automatically completes the command `user`. Pressing the tab key again will display a list of all the `user` commands.
- If you specify an option value that contains a space, enclose the value in double quotes.

8. When you are done, type `exit` to exit the Administration Utility.

Setting up Command Line Interface (CLI) properties in an HTTPS-enabled server environment

If using self-signed certificates , make sure to import them to your local machine.

1. Import your self-signed certificates.
   For example:

```
keytool -importkeystore -srckeystore "C:\Program
Files\Precisely\Spectrum\server\conf\certs\node-keystore.p12"
     -destkeystore "C:\Program
Files\Precisely\Spectrum\server\conf\certs\truststore.p12"
-deststoretype pkcs12
```

2. In the same directory where your CLI executable is located, create a file called: `cli.properties`.
   Here is a sample of the file contents:

```
# sample properties
spectrum.encryption.keystoreType=pkcs12
spectrum.encryption.keystore=C:\\Users\\Spectrum\\mycerts\\node-keystore.p12
spectrum.encryption.keystorePassword=p1tn3yb0w3s
spectrum.encryption.keystoreAlias=spectrum
spectrum.encryption.truststoreType=pkcs12
spectrum.encryption.truststore=C:\\Users\\Spectrum\\mycerts\\truststore.p12
spectrum.encryption.truststorePassword=p1tn3yb0w3s
spectrum.encryption.truststoreAlias=spectrum
spectrum.encryption.trustAllHosts=true
spectrum.encryption.trustSelfSigned=false
```

Related concepts

Implementing self-signed certificates on page 54

Spectrum SSL properties offer varying degrees of control of certificate verification through Certificate Authorities (CAs).

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`
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
```

Constraints and Limitations

The following constraints or limitations should be accounted for when using the Administration Utility or Command Line Interface.

- The CLI uses the default encoding of the Java virtual machine (JVM) on the system where the Spectrum Technology Platform is installed.

Audit Log Information

Audit Log Information CLI commands adds JSON activity and count information to the audit log files.

auditlog export

The `auditlog export` command adds a JSON activity log to all audit log files. Times are in `yyyyMMddHHmmss` format. If no specific timeframe is specified, the default is the current day's start date and the time you issued the `auditlog export` command.

Usage

`auditlog export --n` *name* `--v` *value* `--s` *startTime* `--e` *endTime* `--f` *filterBy* `--fw` *filterByWild* `--fa` *filterByAdditional*

Required	Argument	Description
No	`--n` *name*	Specifies the name of the field to use on the activity log. For example: "username."
No	`--v` *value*	Specifies the value that goes with the `name` definition. For example: "admin."
No	`--s` *startTime*	Specifies start time and start date for audit logging. The date format is: `yyyyMMddHHmmss`.
No	`--e` *endTime*	Specifies stop time and end date for audit logging. The date format is: `yyyyMMddHHmmss`.
No	`--f` *filterBy*	Specifies an entity to use for filtering information in the activity log. For example: `username:system`.
No	`--fw` *filterByWild*	Allows you to use the asterisk (*) character to filter the information returned. For example, to search for an object ID containing the string "info," specify `objectID:*info`.
No	`--fa` *filterByAdditional*	Specifies an additional value to use in filtering the information returned. For example, you could use a specific date to restrict the returned information to a calendar day.

Example

This example asks to return results for a 24-hour timebox, for admin-level users.

```
auditlog export --s 20191231000000 --e 20200101000000 --f
userlevel:admin
```

auditlog info (audit log information summary)

The `auditlog info` command adds a JSON count information file to the audit log files. Times are in `yyyyMMddHHmmss` format. If no specific timeframe is specified, the default is the current day's start date and the time you issue the auditlog info command. This command provides multiple filtering options for the data returned. You direct the JSON count file to an output directory of your choice.

Usage

`auditlog info --n` *fieldName* `--s` *startTime* `--e` *endTime* `--f` *filterBy* `--fw` *filterByWild* `--fa` *filterByAdditional* `--ob` *orderBy* `--a` *ascending* `--o`*directory*

Required	Argument	Description
No	`--n` *fieldName*	Specifies the name of the field to include on the auditlog info returned. You can specify more than one field name. For example, "username" and "value" are companion fields, so you may want to include both in your results.
No	`--s` *startTime*	Specifies start time and start date for audit logging. The date format is: `yyyyMMddHHmmss`.
No	`--e` *endTime*	Specifies end time and end date for audit logging. The date format is: `yyyyMMddHHmmss`.
No	`--f` *filterBy*	Specifies a specific entity to use as a results filter. For example, "username:system."
No	`--fw` *filterByWild*	Allows you to use the asterisk (*) character to filter the information returned. For example, to search for an object ID containing the string "info," specify `objectID:*info`.
No	`--fa` *filterByAdditional*	Specifies an additional value to use in filtering the information returned. For example, you could use a specific date to restrict the returned information to a calendar day.
No	`--ob` *orderByType*	Allows you to order the returned information by "loglevel" or "timestamp." The default ordering is by time stamp.
No	`--a` *ascending*	Shows Boolean results in ascending order. The default ordering is true then false if this filter is not specified.
No	`--o` *directory*	Specifies the output directory for the auditlog information.

Example

This example asks to return results for a 24-hour timebox, for an admin-level user, ordered from earliest event to latest event, sending the results to a directory named `c:\Precisely\auditlog_info\results`.

```
auditlog info --s 20191231000000 --e 20200101000000 --f
userlevel:admin --ob timestamp --o
c:\Precisely\auditlog_info\results
```

Spectrum Business Glossary

Spectrum Business Glossary CLI commands list, import and export glossary entities.

Glossaryentity Export

Use this command to export glossary entities from the Spectrum server to a given directory in a CSV format.

Usage

`glossaryentity export --n` *name* `--o` *outputPath* `--d` *delimiter*

Required	Argument	Description
Yes	`--n` *name*	Specify the name of the glossary entity to be exported. **Tip:** If you are unsure of the glossary entity name, you can use the `glossaryentity list` command to get a list of the names.
No	`--o` *outputPath*	Specify the output directory to export the glossary entity. **Note:** If you do not specify this path, the entities are saved to the directory from which you are running the command.
No	`--d` *delimiter*	Specify the delimiter to be used in the export file. The supported delimiters are: `Comma (,) Semicolon (;) Pipe(|) Tab(\t).` Default value is pipe (|). **Note:** Do not use comma as a delimiter if entity description contains commas. Else, import might fail.

Example

This example exports glossary entity "Customer" from Spectrum server to the folder `MyGlossary`. The delimiter to be used in the export file is semicolon.

```
glossaryentity export --n Customer --o D:/Export/MyGlossary
--d ;
```

Glossaryentity Import

Use this command to import glossary entities from a CSV file and create a version 1.0 draft.

Usage

`glossaryentity import --i` *inputPath* `--d` *delimiter*

Required	Argument	Description
Yes	`--i` *inputPath*	Path to the CSV file or the folder containing the CSV files, from which the glossary entities are to be imported. **Note:** If the input path is of a folder, ensure all the files in the folder use the same delimiter. **Note:** Use forward slash in file or folder paths to avoid any issues.
No	`--d` *delimiter*	Specify the delimiter used in import file. The supported delimiters are: `Comma (,) Semicolon (;) Pipe(|) Tab(\t)`. Default value is pipe (|). **Note:** Do not use comma as a delimiter if entity description contains commas. Else, import might fail.

Example

This example imports glossary entities from a CSV file in the folder `MyGlossary`. All the files in the folder use comma as the delimiter.

```
glossaryentity import --i D:/Import/MyGlossary --d ,
```

Sample import file

```
EntityName:CustomerEntity

Description:The customer's information

PropertyName|PropertyDescription|PropertyDataType
```

```
FirstName|First Name|string
LastName|Last Name|string
Phone|Phone Number|Phone
EmailID|Email Address|Email
```

Note: Ensure that all data types used in the import CSV file are there in the semantic types. If not, create it before importing the entity on the server to avoid import failure.

Glossaryentity List

Use this command to view a list of existing glossary entities.

Usage

```
glossaryentity list
```

Example

This example lists all the existing glossary entities.

```
glossaryentity list
```

Data Stewardship

The Data Stewardship CLI command deletes exception records from the repository.

bsm delete exceptions

Use this command to delete exception records from the repository. You can choose to delete exception records produced by a specific job in a dataflow or by all jobs in a dataflow.

Usage

`bsm delete exceptions --n` *name* `--i` *id* `--r` *reports*

Required	Argument	Description
Yes	`--n` *name*	Specifies the dataflow name.

Required	Argument	Description
No	`--i` *id*	Specifies the job ID. Include this argument to delete exception records that were produced by a single job in the dataflow. Omit this argument to delete exception records that were produced by all of the jobs in the dataflow.
No	`--r` *reports*	Specifies whether to remove the data quality reports associated with the exception records. **true** Removes performance data along with the exception records. This is the default setting. **false** Exception records are removed from the repository, but the performance data is retained and still appears on the Data Quality page of the Exception Monitor.

Example

This example removes all exceptions for job ID 24 in the dataflow named "My Dataflow".

```
bsm delete exceptions --n "My Dataflow" --i 24
```

Context Graph

The Administration Utility Command Line Interface (CLI) provides Context Graph (`contextgraph`) commands to manage models, view jobs and job status, and manage model schemas, queries, and metadata.

Command	Description
contextgraph job list on page 283	Returns a list of all Context Graph jobs.
contextgraph job status on page 284	Returns the status of a Context Graph job.
contextgraph model clear on page 284	Removes the contents of a Context Graph model.
contextgraph model delete on page 285	Deletes a Context Graph model.

Command	Description
contextgraph model list on page 285	Lists Context Graph models.
contextgraph query delete on page 286	Delete a query of a model.
contextgraph query list on page 287	List queries of a model.
contextgraph schema copy on page 288	Copies model metadata.
contextgraph schema count entityProperty on page 289	Counts an entity property of a model.
contextgraph schema count relationshipProperty on page 290	Counts a relationship property of a model.
contextgraph schema delete entityProperty on page 290	Delete a Context Graph model property.
contextgraph schema delete entityType on page 291	Delete a model entity type.
contextgraph schema delete relationshipLabel on page 292	Deletes a relationship label from a model.
contextgraph schema delete relationshipProperty on page 293	Deletes a relationship property from a Context Graph model.
contextgraph schema export on page 294	Exports a model.
contextgraph schema import on page 295	Imports a model.
contextgraph schema importLogicalModel on page 295	Imports a logical model from Discovery into Context Graph.
contextgraph schema list all on page 296	List entity types, relationship labels and total counts for a model.
contextgraph schema list entityProperties on page 297	Lists entity properties for a model.
contextgraph schema list entityTypes on page 298	Lists entity types for a model.
contextgraph schema list relationshipLabels on page 298	Lists relationship labels for a model.
contextgraph schema list relationshipProperties on page 299	Lists entity properties for a model.
contextgraph schema modify indexType on page 300	Changes the index type of a model.

Command	Description
contextgraph schema rename entityProperty on page 301	Rename a model property.
contextgraph schema rename entityType on page 302	Rename an entity type for a model.
contextgraph schema rename relationshipLabel on page 302	Rename a relationship label.
contextgraph schema rename relationshipProperty on page 303	Rename a model property.

contextgraph job list

Returns a list of all Context Graph jobs.

Use the `contextgraph job list` command to return a list of all Context Graph jobs with or without date and time specifications.

Usage

`contextgraph job list --f` *from datetime* `--t` *to datetime*

Required	Argument	Description
No	`--f` *from datetime*	If you want to see the list for a specific date and time range, specify the starting date and time for the range, in the format 'MM-dd-yyyy HH:mm:ss'. For example, December 31, 2014 1:00 PM would be specified as `'12-31-2014 13:00:00'`. When you specify a date and time range, the list will include jobs that started execution on or after the date you specified in the `--f` argument and before the date you specify in the `--t` argument. If you omit this argument the list will include jobs that started execution on the current date.
No	`--t` *to datetime*	If you want to see the list for a specific date and time range, specify the ending date and time for the range, in the format `'`*MM-dd-yyyy HH:mm:ss*`'`. For example, December 31, 2014 1:00 PM would be specified as `'12-31-2014 13:00:00'`.

Required	Argument	Description
		When you specify a date and time range, the list will include jobs that started execution on or after the date you specified in the `--f` argument and before the date you specify in the `--t` argument. If you omit this argument the list will include jobs that started execution on or after the date specified in the `--f` argument.

Example

This example lists all Context Graph jobs run on or after January 1, 2010 at 00:00:00.

```
contextgraph job list --f '01-01-2010 00:00:00'
```

contextgraph job status

Returns the status of a Context Graph job.

Use the `contextgraph job status` command to return the status of a Context Graph job.

Usage

`contextgraph job status --id` *jobID*

Required	Argument	Description
Yes	`--id` *jobID*	Specifies the ID of the Context Graph job.

Example

This example returns the status of Context Graph job 24.

```
contextgraph job status --id 24
```

contextgraph model clear

Removes the contents of a Context Graph model.

Use the `contextgraph model clear` command to remove the contents of a Context Graph model but leave it and its metadata in place.

Usage

`contextgraph backup clear --m` *model*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose contents you want to clear.

Example

This example clears a model called CustomerDB_032018.

```
contextgraph model clear --m CustomerDB_032018
```

contextgraph model delete

Deletes a Context Graph model.

Use the `contextgraph model delete` command to delete a specific Context Graph model.

Usage

`contextgraph model delete --m` *model*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model you want to delete.

Example

This example deletes a model called PersonalBanking.

```
contextgraph model delete --m PersonalBanking
```

contextgraph model list

Lists Context Graph models.

Use the `contextgraph model list` command to return a list of all Context Graph models as well as counts for entities and relationships for each model.

Usage

`contextgraph model list --c` *counts*

Required	Argument	Description
No	`--c` *counts*	Specifies whether to include counts for entities and relationships, where *counts* is one of the following: **true** Includes counts. This is the default setting. **false** Does not include counts.

Example

This example lists all Context Graph models and provides counts for entities and relationships for each model.

```
contextgraph model list --c true
```

contextgraph query delete

Delete a query of a model.

Use the `contextgraph query delete` command to delete a query of a model.

Usage

`contextgraph query delete --m` *model* `--q` *query*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose query you want to delete.
Yes	`--q` *query*	Name of the query to be deleted.

Example

This example deletes the query called PersonalQuery from a model called PersonalLending.

```
contextgraph query delete --m PersonalLending --q
PersonalQuery
```

contextgraph query list

List queries of a model.

Use the `contextgraph query list` command to list queries of a model. You can include the optional arguments to list specific types of queries. All queries of a model are returned if none of the arguments (`--c`, `--e`, `--v`, or `--s`) are included as true.

Usage

`contextgraph query list --m` ***model*** `--c --e --v --s`

Required	Argument	Description
Yes	`--m` ***model***	Specifies the name of the model whose queries you want to list.
No	`--c` ***custom***	Returns custom script queries created in the Read from Model and Query Model stages. The argument may be set to true or false: **true** Returns all custom queries. This is the default value if this argument is included. **false** Does not return custom queries. This is the default if this argument is omitted and any of the other types (`--e`, `--v`, or `--s`) is included as true.
No	`--e` ***entity***	Returns Entity Browser queries created in Context Graph Visualization. The argument may be set to true or false: **true** Returns all Context Graph Browser queries. This is the default value if this argument is included. **false** Does not return Context Graph Browser queries. This is the default if this argument is omitted and any of the other types (`--c`, `--v`, or `--s`) is included as true.
No	`--v` ***visual***	Returns Visual Query Builder queries created in Context Graph Visualization. The argument may be set to true or false: **true** Returns all *visual* queries. This is the default value if this argument is included. **false**

Required	Argument	Description
		Does not return *visual* queries. This is the default if this argument is omitted and any of the other types (`--c`, `--e`, or `--s`) is included as true.
No	`--s` *shared*	Returns queries created in the Read from Model and Query Model stages; and in Map, Table, and the Canvas Query tool in Context Graph Visualization. The argument may be set to true or false: **true** Returns all *shared* queries. This is the default value if this argument is included. **false** Does not return *shared* queries. This is the default value if this argument is omitted and any of the other types (`--c`, `--e`, or `--v`) is included as true.

Example

This example lists all queries of the Artists model.

```
contextgraph query list --m Artists
```

Example

This example lists all visual queries of the Artists model.

```
contextgraph query list --m Artists --v
```

contextgraph schema copy

Copies model metadata.

Use the `contextgraph schema copy` command to copy Context Graph model metadata, and optionally its queries and themes.

Usage

`contextgraph schema copy --m` *model* `--nm` *newmodel* `--cq` *copyqueries* `--ct` *copythemes*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose schema you want to copy.
Yes	`--nm` *newmodel*	Specifies the name of the new model.

Required	Argument	Description
No	`--cq` *copyqueries*	Specifies whether to copy any saved queries from the old model into the new model, where *copyqueries* is one of the following: **true** Copies queries. This is the default setting. **false** Does not copy queries.
No	`--ct` *copythemes*	Specifies whether to copy any themes from the old model into the new model, where *copythemes* is one of the following: **true** Copies themes. This is the default setting. **false** Does not copy themes.

Example

This example copies the schema from a model called PersonalLending from the default backup folder and names the copy PersonalLending_New. It also copies any themes associated with the old model but does not copy any queries associated with the old model.

```
contextgraph schema copy --m PersonalLending --nm
PersonalLending_New --cm true --cq false --ct true
```

contextgraph schema count entityProperty

Counts an entity property of a model.

Use the `contextgraph schema count entityProperty` command to return the count for an entity property in a Context Graph model.

Usage

`contextgraph schema count entityProperty --m` *model* `--e` *entitytype* `--p` *property*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose property you want to count.
No	`--e` *entitytype*	Specifies the entity type on which to filter results.
Yes	`--p` *property*	Specifies the property to be counted.

Example

This example returns the count for the `Title` property on the `Single` entity in the `PopularMusic` model.

```
contextgraph schema count entityProperty --m PopularMusic --e
Single --p Title
```

contextgraph schema count relationshipProperty

Counts a relationship property of a model.

Use the `contextgraph schema count relationshipProperty` command to count a relationship property in a Context Graph model.

Usage

`contextgraph schema count relationshipProperty --m` *model* `--r` *relationshiplabel* `--p` *property*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose relationship you want to count.
No	`--r` *relationshipLabel*	Specifies the relationship label on which to filter results.
Yes	`--p` *property*	Specifies the property to be counted.

Example

This example returns the count for the `Date` property on the `Recorded` relationship in the `PopularMusic` model.

```
contextgraph schema count relationshipProperty --m
PopularMusic --r Recorded --p Date
```

contextgraph schema delete entityProperty

Delete a Context Graph model property.

Use the `contextgraph schema delete entityProperty` command to delete a Context Graph model property.

Usage

`contextgraph schema delete entityProperty --m` *model* `--e` *entityType* `--p` *property* `--w` *waitForComplete*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose entity type property you want to delete.
No	`--e` *entityType*	Specifies the target entity type; includes all entity types if not specified.
Yes	`--p` *property*	Specifies the property you want to delete.
No	`--w` *waitForComplete*	Specifies whether to wait for jobs to complete in a synchronous mode, where *waitForComplete* is one of the following: **true** Waits for jobs to complete. **false** Does not wait for jobs to complete. This is the default setting.

Example

This example deletes the property HireDate from a model called Staff and an entity type of EmployeeName.

```
contextgraph schema delete entityProperty --m Staff --e
EmployeeName --p HireDate
```

contextgraph schema delete entityType

Delete a model entity type.

Use the `contextgraph schema delete entityType` command to delete a Context Graph model entity type. It optionally specifies whether to complete jobs in synchronous mode.

Usage

`contextgraph schema delete entityType --m` *model* `--e` *entityType* `--w` *waitForComplete*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose entity type you want to delete.
Yes	`--e` *entityType*	Specifies the type of entity to be deleted.

Required	Argument	Description
No	`--w` *waitForComplete*	Specifies whether to wait for jobs to complete in a synchronous mode, where *waitForComplete* is one of the following: **true** Waits for jobs to complete. **false** Does not wait for jobs to complete. This is the default setting.

Example

This example deletes the entity type Employee from a model called PersonalLending.

```
contextgraph schema delete entityType --m PersonalLending --e
Employee
```

contextgraph schema delete relationshipLabel

Deletes a relationship label from a model.

Use the `contextgraph schema delete relationshipLabel` command to delete a relationship label from a model. You may optionally choose wait for other jobs in synchronous mode.

Usage

`contextgraph schema delete relationshipLabel --m` *model* `--r` *relationshipLabel* `--s` *sourceEntityType* `--t` *targetEntityType* `--w` *waitForComplete*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose relationship label you want to delete.
Yes	`--r` *relationshipLabel*	Specifies the relationship label to be deleted.
No	`--s` *sourceEntityType*	Specifies the type of source entity.
No	`--t` *targetEntityType*	Specifies the type of target entity.
No	`--w` *waitForComplete*	Specifies whether to wait for jobs to complete in a synchronous mode, where *waitForComplete* is one of the following: **true** Waits for jobs to complete. **false**

Required	Argument	Description
		Does not wait for jobs to complete. This is the default setting.

Example

This example deletes the relationshipLabel Hired from a model called Staff.

```
contextgraph schema delete relationshipLabel --m Staff --r
Hired
```

contextgraph schema delete relationshipProperty

Deletes a relationship property from a Context Graph model.

Use the `contextgraph schema delete relationshipProperty` command to delete a Context Graph model relationship property.

Usage

`contextgraph schema delete relationshipProperty --m` *model* `--r` *relationshipLabel* `--p` *property* `--s` *sourceEntityType* `--t` *targetEntityType* `--w` *waitForComplete*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose entity type or relationship label property you want to delete.
No	`--r` *relationshipLabel*	Specifies the target relationship label; includes all relationship labels if not specified.
Yes	`--p` *property*	Specifies the property you want to delete.
No	`--s` *sourceEntityType*	Specifies the type of source entity.
No	`--t` *targetEntityType*	Specifies the type of target entity.
No	`--w` *waitForComplete*	Specifies whether to wait for jobs to complete in a synchronous mode, where *waitForComplete* is one of the following: **true** Waits for jobs to complete. **false** Does not wait for jobs to complete. This is the default setting.

Example

This example deletes the property HireDate from a model called Staff and a relationship label of Hired.

```
contextgraph schema delete relationshipProperty --m Staff --r
Hired --p HireDate
```

contextgraph schema export

Exports a model.

Use the `contextgraph schema export` command to export a Context Graph model and its metadata, and optionally its queries and themes. If you do not specify a path to where you would like the model exported, the system export the model to the current working directory using the name you specify.

Usage

`contextgraph schema export --m` *model* `--f` *file* `--cq` *copyqueries* `--ct` *copythemes*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model you want to export.
Yes	`--p` *path*	(Deprecated) Specifies the path where you want to save the export folder. This path is relative to where you have installed the Spectrum Technology Platform server.
No	`--f` *file*	Specifies the path where you want to save the export folder. This path is relative to where you have installed the Spectrum Technology Platform Administration Utility.
No	`--cq` *copyqueries*	Specifies whether to export any saved queries, where *copyqueries* is one of the following: **true** Exports queries. This is the default setting. **false** Does not export queries.
No	`--ct` *copythemes*	Specifies whether to export any themes with the schema. **true** Exports themes. This is the default setting. **false** Does not export themes.

Example

This example exports the schema for a model called Fraud_2015 to the `GraphModels` directory on the C drive.

```
contextgraph schema export --m Fraud_2015 --f C:\GraphModels
--cm false --cq true -ct true
```

contextgraph schema import

Imports a model.

Use the `contextgraph schema import` command to import a Context Graph model, its metadata, and its queries. If you do not specify a path where you would like the model imported, the system will look for a file of the name you specify in the current working directory.

Usage

`contextgraph schema import --m` *model* `--f` *file*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose schema you want to import.
No	`--p` *path*	(Deprecated) Specifies the path for the location of the model whose schema you are importing. This path is relative to where you have installed the Spectrum Technology Platform server.
No	`--f` *file*	Specifies the path for the location of the model whose schema you are importing. This path is relative to where you have installed the Spectrum Technology Platform Administration Utility.

Example

This example imports the schema for a model called Fraud_2015 from the `GraphModels` directory on the C drive.

```
contextgraph schema import --m Fraud_2015 --f C:\GraphModels
```

contextgraph schema importLogicalModel

Imports a logical model from Discovery into Context Graph.

Use the `contextgraph schema importLogicalModel` command to import a Metadata Insights Logical Model into Context Graph.

Usage

```
contextgraph schema importLogicalModel --m model --n logicalModelName
```

Required	Argument	Description
Yes	`--m` *model*	Specifies the name you would like to give the model in Context Graph.
No	`--n` *logicalModelName*	Specifies the name of the Discovery model whose schema you are importing.

Example

This example imports a Discovery model called Insured and names it Insured2018.

```
contextgraph schema importLogicalModel --m Insured2018 --n
Insured
```

contextgraph schema list all

List entity types, relationship labels and total counts for a model.

Use the basic `contextgraph schema list all` command to return a list of entity types, relationship labels and total counts for a model. Add the *verbose* argument to include entity properties, relationship label connections, relationship properties, and indexed properties.

Usage

```
contextgraph schema list all --m model --v verbose
```

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose schema information you want to list.
No	`--v` *verbose*	Specifies whether to include verbose output, where *verbose* is one of the following: **true** Includes verbose output. **false** Does not include verbose output. This is the default setting.

Example

This example lists all relationship properties for a model named PersonalBanking and does not include verbose output.

```
contextgraph schema list all --m PersonalBanking
```

contextgraph schema list entityProperties

Lists entity properties for a model.

Use the `contextgraph schema list entityProperties` command to return a list of all entity properties for a Context Graph model.

Usage

`contextgraph schema list entityProperties --m` *model* `--e` *entityType* `--i` *indexedOnly*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose entity properties you want to list.
No	`--e` *entityType*	Limits the results to the specified entity type.
No	`--i` *indexedOnly*	Specifies whether to limit the results to indexed properties only, where *indexedOnly* is one of the following: **true** Limits the results. This is the default setting. **false** Does not limit the results.

Example

This example lists all entity properties for a model named PersonalBanking, filters the results to include just the Customer type, and filters the results to indexed properties only.

```
contextgraph schema list entityProperties --m PersonalBanking
--e Customer
```

contextgraph schema list entityTypes

Lists entity types for a model.

Use the `contextgraph schema list entityTypes` command to return a list of all entity types for a Context Graph model.

Usage

`contextgraph schema list entityTypes --m` *model*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose entity types you want to list.

Example

This example lists all entity types for a model named Fraud.

```
contextgraph schema list entityTypes --m Fraud
```

Related information

The Neo4j Graph Algorithms User Guide

contextgraph schema list relationshipLabels

Lists relationship labels for a model.

Use the `contextgraph schema list relationshipLabels` command to return a list of all relationship labels for a Context Graph model.

Usage

`contextgraph schema list relationshipLabels --m` *model* `--s` *sourceEntityType* `--t` *targetEntityType* `--c` *showConnections*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose relationship labels you want to return.
No	`--s` *sourceEntityType*	Specifies the type of source entity.
No	`--t` *targetEntityType*	Specifies the type of target entity.

Required	Argument	Description
No	`--c` *showConnections*	Specifies whether to show source and target entity types, where *showConnections* is one of the following: **true** Shows connections. **false** Does not show connections. This is the default setting. .

Example

This example returns a list of relationship labels with source and target entity types for a model called June2017 with a source entity type of Customer and a target entity type of AccountType.

contextgraph schema list relationshipLabels --m June2017 --s Customer --t AccountType --c

contextgraph schema list relationshipProperties

Lists entity properties for a model.

Use the `contextgraph schema list relationshipProperties` command to return a list of all entity properties for a Context Graph model.

Usage

`contextgraph schema list relationshipProperties --m` *model* `--r` *relationshipLabel*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose relationship properties you want to list.
No	`--r` *relationshipLabel*	Filters the results to the specified relationship label type.

Example

This example lists all relationship properties for a model named PrivateBanking and filters the results to include just the Current type.

```
contextgraph schema list relationshipProperties --m
PrivateBanking --r Current
```

contextgraph schema modify indexType

Changes the index type of a model.

Use the `contextgraph schema modify indexType` command to change the index type of a Context Graph model property.

Usage

`contextgraph schema modify indexType --m` *model* `--p` *property* `--i` *index* **--w** *waitForComplete*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose entity type you want to modify.
Yes	`--p` *property*	Specifies the property you want to index.
Yes	`--i` *index*	Specifies the property index. **NONE** Removes the property index. **EXACT** Sets the property index to exact. **CASE_INSENSITIVE** Sets the property index to case insensitive.
No	`--w` *waitForComplete*	Specifies whether to wait for jobs to complete in a synchronous mode, where *waitForComplete* is one of the following: **true** Waits for jobs to complete. **false** Does not wait for jobs to complete. This is the default setting.

Example

This example changes the index type for a property called HireDate in a model called Staff to exact.

```
contextgraph schema modify indexType --m Staff --p HireDate
--i EXACT
```

contextgraph schema rename entityProperty

Rename a model property.

Use the `contextgraph schema rename entityProperty` command to rename a Context Graph model property.

Usage

`contextgraph schema rename entityProperty --m` *model* `--e` *entityType* `--p` *property* `--np` *newProperty* `--w` *waitForComplete*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose entity property you want to rename.
No	`--e` *entityType*	Specifies the target entity type; includes all entity types if not specified.
Yes	`--p` *property*	Specifies the property you want to rename.
Yes	`--np` *newProperty*	Specifies the new property name.
No	`--w` *waitForComplete*	Specifies whether to wait for jobs to complete in a synchronous mode, where *waitForComplete* is one of the following: **true** Waits for jobs to complete. **false** Does not wait for jobs to complete. This is the default setting.

Example

This example renames the property HireDate to Start Date in a model called Staff with a relationship label of Hired.

```
contextgraph schema rename entityProperty --m Staff --r Hired
--p HireDate --np StartDate
```

contextgraph schema rename entityType

Rename an entity type for a model.

Use the `contextgraph schema rename entityType` command to rename an entity type in a Context Graph model.

Usage

`contextgraph schema rename entityType --m` *model* `--e` *entityType* `--ne` *newEntityType* `--w` *waitForComplete*

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose entity type you want to rename.
Yes	`--e` *entityType*	Specifies the type of entity to be renamed.
Yes	`--ne` *newEntityType*	Specifies the new entity type.
No	`--w` *waitForComplete*	Specifies whether to wait for jobs to complete in a synchronous mode, where *waitForComplete* is one of the following: **true** Waits for jobs to complete. **false** Does not wait for jobs to complete. This is the default setting.

Example

This example renames an entity type from Employee to Staff in a model called PersonalLending.

```
contextgraph schema rename entityType --m PersonalLending --e
Employee --ne Staff
```

contextgraph schema rename relationshipLabel

Rename a relationship label.

Use the `contextgraph schema rename relationshipLabel` command to rename a Context Graph model relationship label.

Usage

```
contextgraph schema rename relationshipLabel --m model --r relationshipLabel
--nr newRelationshipLabel --s sourceEntityType --t targetEntityType --w waitForComplete
```

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose relationship label you want to rename.
Yes	`--r` *relationshipLabel*	Specifies the relationship label you want to rename.
Yes	`--nr` *newRelationshipLabel*	Specifies the new relationship label name.
No	`--s` *sourceEntityType*	Specifies the type of source entity.
No	`--t` *targetEntityType*	Specifies the type of target entity.
No	`--w` *waitForComplete*	Specifies whether to wait for jobs to complete in a synchronous mode, where *waitForComplete* is one of the following: **true** Waits for jobs to complete. **false** Does not wait for jobs to complete. This is the default setting.

Example

This example renames the relationshipLabel Hired to Employed in a model called Staff.

```
contextgraph schema rename relationshipLabel --m Staff --r
Hired --nr Employed
```

contextgraph schema rename relationshipProperty

Rename a model property.

Use the `contextgraph schema rename relationshipProperty` command to rename a property in a Context Graph model.

Usage

```
contextgraph schema rename relationshipProperty --m model --r relationshipLabel
--p property --np newProperty --s sourceEntityType --t targetEntityType --w waitForComplete
```

Required	Argument	Description
Yes	`--m` *model*	Specifies the name of the model whose relationship property you want to rename.
No	`--r` *relationshipLabel*	Specifies the target relationship label.
Yes	`--p` *property*	Specifies the property you want to rename.
Yes	`--np` *newProperty*	Specifies the new property name.
No	`--s` *sourceEntityType*	Specifies the type of source entity.
No	`--t` *targetEntityType*	Specifies the type of target entity.
No	`--w` *waitForComplete*	Specifies whether to wait for jobs to complete in a synchronous mode, where *waitForComplete* is one of the following: **true** Waits for jobs to complete. **false** Does not wait for jobs to complete. This is the default setting.

Example

This example renames the relationship property HireDate to Start Date in a model called Staff with a relationship label of Hired.

```
contextgraph schema rename relationshipProperty --m Staff --r
Hired --p HireDate --np StartDate
```

Data Sources

Data source CLI commands manage connection between the Spectrum Technology Platform and data sources.

FTP

- **data source ftp add** on page 305
- **data source ftp delete** on page 306
- **data source ftp list** on page 306
- **data source ftp test** on page 307
- **data source ftp update** on page 307

SFTP

- **data source sftp add** on page 308
- **data source sftp delete** on page 309
- **data source sftp list** on page 309
- **data source sftp test** on page 310
- **data source sftp update** on page 310

JDBC Database

Connections

- **dbconnection add** on page 311
- **dbconnection delete** on page 312
- **dbconnection export** on page 313
- **dbconnection import** on page 313
- **dbconnection list** on page 314
- **dbconnection test** on page 314
- **dbconnection update** on page 315

Drivers

- **dbdriver delete** on page 316
- **dbdriver export** on page 316
- **dbdriver import** on page 317
- **dbdriver list** on page 318

FTP

data source ftp add

The `data source ftp add` command creates a connection between Spectrum Technology Platform and an FTP server.

Usage

`data source ftp add --n` *ConnectionName* `--h` *Host* `--o` *Port* `--u` *Username* `--p` *Password*

Required	Argument	Description
Yes	`--n` *ConnectionName*	Specifies the name for the connection. The name can be anything you choose.
Yes	`--h` *Host*	Specifies the host name or IP address of the FTP server.
No	`--o` *Port*	Specifies the network port to use for communication with the FTP server.

Required	Argument	Description
No	`--u` *Username*	The user name to use to connect to the FTP server, if required.
No	`--p` *Password*	The password to use to connect to the FTP server, if required.

Example

This example creates a connection to the FTP server named MyFTPServer.

```
data source ftp add --n NorthernRegionCustomers --h
MyFTPServer --u ExampleUsername --p Example123
```

data source ftp delete

The `data source ftp delete` command deletes a connection between Spectrum Technology Platform and an FTP server.

Usage

`data source ftp delete --n` *ConnectionName*

Required	Argument	Description
Yes	`--n` *ConnectionName*	Specifies the name of the connection you want to delete. To view a list of connections, use the `data source ftp list` command.

Example

This example deletes a connection named NorthernRegionCustomers.

```
data source ftp delete --n NorthernRegionCustomers
```

data source ftp list

The `data source ftp list` command returns a list of the FTP connections defined on the Spectrum Technology Platform server.

Usage

```
data source ftp list
```

data source ftp test

The `data source ftp test` command tests a connection between Spectrum Technology Platform and an FTP server.

Usage

`data source ftp test --n` *ConnectionName*

Required	Argument	Description
Yes	`--n` *ConnectionName*	Specifies the name of the connection you want to test. To view a list of connections, use the `data source ftp list` command.

Example

This example test the connection NorthernRegionCustomers.

```
data source ftp test --n NorthernRegionCustomers
```

data source ftp update

The `data source ftp update` command modifies a connection between Spectrum Technology Platform and an FTP server.

Usage

`data source ftp update --n` *ConnectionName* `--h` *Host* `--o` *Port* `--u` *Username* `--p` *Password*

Required	Argument	Description
Yes	`--n` *ConnectionName*	Specifies the name for the connection you want to modify. To view a list of connections, use the `data source ftp list` command.
Yes	`--h` *Host*	Specifies the host name or IP address of the FTP server.
No	`--o` *Port*	Specifies the network port to use for communication with the FTP server.
No	`--u` *Username*	The user name to use to connect to the FTP server, if required.
No	`--p` *Password*	The password to use to connect to the FTP server, if required.

Example

This example modifies an FTP connection named NorthernRegionCustomers. It changes the host to `MyFTPServer2`.

```
data source ftp update --n NorthernRegionCustomers --h
MyFTPServer2
```

SFTP

data source sftp add

The `data source sftp add` command creates a connection between Spectrum Technology Platform and an SFTP server.

Usage

`data source sftp add --n` *ConnectionName* `--h` *Host* `--o` *Port* `--s` *strictHostCheck* `--u` *Username* `--a` *key-based authentication* `--k` *privateKeyFile* `--e` *passphrase* `--f` *knownHostFile*

Required	Argument	Description
Yes	`--n` *ConnectionName*	Specifies the name for the connection. The name can be anything you choose.
Yes	`--h` *Host*	Specifies the host name or IP address of the SFTP server.
No	`--o` *Port*	Specifies the network port to use for communication with the SFTP server. Default is 22.
No	`--s` *strictHostCheck*	Specifies if you want Strict Host Key Checking enabled. Default is "false"
No	`--u` *Username*	The user name to use to connect to the SFTP server, if required.
No	`--a` *key-based authentication*	Specifies if authentication is a Password or is Key-Based. Default is Password.
No	`--k` *privateKeyFile*	Specifies the private key file path
No	`--e` *passphrase*	Specifies the Passphrase set with private key generation.
No	`--f` *knownHostFile*	Specifies location of the file that maintains known hosts details.
No	`--p` *Password*	The password to use to connect to the SFTP server, if required.

Example

This example creates a connection to the SFTP server named MySFTPServer with a `key-based` authentication.

```
data source sftp add --n NorthernRegionCustomers --h
MySFTPServer --o 22 --u ExampleUserName --a Key-Based --k
ExampleKeyFile --e ExamplePassphrase
```

Example 2:

This example creates a connection to the SFTP server named MySFTPServer when **Authentication type** is Password.

```
data source sftp add --n NorthernRegionCustomers --h
MySFTPServer --o 22 --u ExampleUserName --a Password --p
Example123
```

data source sftp delete

The `data source sftp delete` command deletes a connection between Spectrum Technology Platform and an SFTP server.

Usage

`data source sftp delete --n` *ConnectionName*

Required	Argument	Description
Yes	`--n` *ConnectionName*	Specifies the name of the connection you want to delete. To view a list of connections, use the `data source sftp list` command.

Example

This example deletes a connection named NorthernRegionCustomers.

```
data source sftp delete --n NorthernRegionCustomers
```

data source sftp list

The `data source sftp list` command returns a list of the SFTP connections defined on the Spectrum Technology Platform server.

Usage

```
data source sftp list
```

data source sftp test

The `data source sftp test` command tests a connection between Spectrum Technology Platform and an SFTP server.

Usage

`data source sftp test --n` *ConnectionName*

Required	Argument	Description
Yes	`--n` *ConnectionName*	Specifies the name of the connection you want to test. To view a list of connections, use the `data source sftp list` command.

Example

This example test the connection NorthernRegionCustomers.

```
data source sftp test --n NorthernRegionCustomers
```

data source sftp update

The `data source sftp update` command modifies a connection between Spectrum Technology Platform and an SFTP server.

Usage

`data source sftp update --n` *ConnectionName* `--h` *Host* `--o` *Port* `--s` *strictHostCheck* `--u` *Username* `--a` *key-based authentication* `--k` *privateKeyFile* `--e` *passphrase* `--f` *knownHostFile*

Required	Argument	Description
Yes	`--n` *ConnectionName*	Specifies the name for the connection. The name can be anything you choose.
Yes	`--h` *Host*	Specifies the host name or IP address of the SFTP server.
No	`--o` *Port*	Specifies the network port to use for communication with the SFTP server. Default is 22.
No	`--s` *strictHostCheck*	Specifies if you want Strict Host Key Checking enabled. Default is "false"
No	`--u` *Username*	The user name to use to connect to the SFTP server, if required.
No	`--a` *key-based authentication*	Specifies if authentication is a Password or is Key-Based. Default is Password.

Required	Argument	Description
No	`--k` *privateKeyFile*	Specifies the private key file path
No	`--e` *passphrase*	Specifies the Passphrase set with private key generation.
No	`--f` *knownHostFile*	Specifies location of the file that maintains known hosts details.
No	`--p` *Password*	The password to use to connect to the SFTP server, if required.

Example

This example modifies an SFTP connection named `NorthernRegionCustomers`. It changes the host to `HostServer2`.

```
data source sftp update --n ConnectionName --h HostServer2
--o Port --s strictHostCheck --u Username --a key-based
authentication --k privateKeyFile --e passphrase --f
knownHostFile
```

JDBC Database

Connections

dbconnection add

The `dbconnection add` command creates a connection between Spectrum Technology Platform and a database.

Usage

`dbconnection add --n` *ConnectionName* `--d` *Driver* `--h` *Host* `--o` *Port* `--i` *Instance* `--u` *Username* `--p` *Password* `--l` "*property*:*value*"

Required	Argument	Description
Yes	`--n` *ConnectionName*	Specifies the name for the connection. The name can be anything you choose.
Yes	`--d` *Driver*	Specifies the driver for the type of database you want to connect to. To view a list of database drivers available on your server, use the `dbdriver list` command.
Yes	`--h` *Host*	Specifies the host name or IP address of the database server.

Required	Argument	Description
No	`--o` *Port*	Specifies the network port to use for communication with the database server.
No	`--i` *Instance*	Specifies the database instance to connect to.
No	`--u` *Username*	The user name to use to connect to the database, if required.
No	`--p` *Password*	The password to use to connect to the database, if required.
No	`--l` "*property*:*value*"	Specifies a comma-separated list of connection property and value pairs for the driver. To view the list of valid properties for a driver, open Spectrum Management Console, go to **Resources** > **Data Sources**, then click the **Drivers** tab. Select the driver you want then click the Edit button to view its connection properties.

Example

This example creates a connection to a database located on the host MyServer. The name of the connection will be NorthernRegionCustomers. It will use the driver ExampleSQLDriver which takes two connection properties: ExampleProp1, which is given a value of 123, and ExampleProp2, which is given a value of 456.

```
dbconnection add --n NorthernRegionCustomers --d
ExampleSQLDriver --h MyServer --l
"ExampleProp1:123,ExampleProp2:456"
```

dbconnection delete

The `dbconnection delete` command deletes a connection between Spectrum Technology Platform and a database.

Usage

`dbconnection delete --n` *ConnectionName*

Required	Argument	Description
Yes	`--n` *ConnectionName*	Specifies the name of the connection you want to delete. To view a list of connections, use the `dbconnection list` command.

Example

This example deletes a connection named NorthernRegionCustomers.

```
dbconnection delete --n NorthernRegionCustomers
```

dbconnection export

The `dbconnection export` command exports a database connection definition to a JSON file.

Usage

`dbconnection export --n` *ConnectionName* `--o` *OutputDirectory*

Required	Argument	Description
Yes	`--n` *ConnectionName*	Specifies the name of the database connection you want to export. If the connection name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the connection name you can use the `dbbconnection list` command to get a list of the connection names.
No	`--o` *OutputDirectory*	Specifies the directory to which you want to export the database connection. The path you specify here is relative to the directory where you are running the Administration Utility. If you omit this argument, the database connection is exported to the directory containing the Administration Utility.

Example

This example exports the database connection named "My Connection" to a folder named `exported` which is a subfolder in the location where you have installed the Administration Utility.

```
dbconnection export --n "My Connection" --o exported
```

dbconnection import

The `dbconnection import` command imports a database connection definition file into the server. Database connection definition files are created by exporting a database connection from the server using the `dbconnection export` command. You can only import database connections that were exported from the same version of Spectrum Technology Platform.

Usage

`dbconnection import --f` *DatabaseConnectionFile* `--u` *TrueOrFalse*

Required	Argument	Description
Yes	`--f` *DatabaseConnectionFile*	Specifies the database connection file you want to import. Connection files have a `.json` extension. Directory paths you specify here are relative to the location where you are running the Administration Utility.

Required	Argument	Description
No	`--u` *TrueOrFalse*	Specifies whether to overwrite the existing database connection if a connection with the same name is already on the server, where *TrueOfFalse* is one of the following: **true** If there is a database connection on the server with the same name as the one you are importing, the connection on the server will be overwritten. This is the default setting. **false** If there is a database connection on the server with the same name as the one you are importing, the connection will not be imported.

Example

This example imports the database connection definition named `MyDatabaseConnection.json` which is located in a subfolder named `exported` in the location where you are running the Administration Utility.

```
dbconnection import --f exported\MyDatabaseConnection.json
```

dbconnection list

The `dbconnection list` command returns a list of the database connections defined on the Spectrum Technology Platform server.

Usage

```
dbconnection list
```

dbconnection test

The `dbconnection test` command tests a connection between Spectrum Technology Platform and a database.

Usage

`dbconnection test --n` *ConnectionName*

Required	Argument	Description
Yes	`--n` *ConnectionName*	Specifies the name of the connection you want to test. To view a list of connections, use the `dbconnection list` command.

Example

This example test the connection NorthernRegionCustomers.

```
dbconnection test --n NorthernRegionCustomers
```

dbconnection update

The `dbconnection update` command modifies a connection between Spectrum Technology Platform and a database.

Usage

`dbconnection update --n` *ConnectionName* `--d` *Driver* `--h` *Host* `--o` *Port* `--i` *Instance* `--u` *Username* `--p` *Password* `--l` "*property*:*value*"

Required	Argument	Description
Yes	`--n` *ConnectionName*	Specifies the name for the connection you want to modify. To view a list of connections, use the `dbconnection list` command.
Yes	`--d` *Driver*	Specifies the driver for the type of database you want to connect to. To view a list of database drivers available on your server, use the `dbdriver list` command.
Yes	`--h` *Host*	Specifies the host name or IP address of the database server.
No	`--o` *Port*	Specifies the network port to use for communication with the database server.
No	`--i` *Instance*	Specifies the database instance to connect to.
No	`--u` *Username*	The user name to use to connect to the database, if required.
No	`--p` *Password*	The password to use to connect to the database, if required.
No	`--l` "*property*:*value*"	Specifies a comma-separated list of connection property and value pairs for the driver. To view the list of valid properties for a driver, open Spectrum Management Console, go to **Resources** > **Data Sources**, then click the **Drivers** tab. Select the driver you want then click the Edit button to view its connection properties.

Example

This example modifies a database connection named NorthernRegionCustomers. It changes the driver to `MSSQLServer2`, changes the host to `MyServer2`, and changes the instance to `MyInstance2`.

```
dbconnection update --n NorthernRegionCustomers --d
MSSQLServer2 --h MyServer2 --i MyInstance2
```

Drivers

dbdriver delete

The `dbdriver delete` command deletes a JDBC driver definition. It does not delete driver files, only the definition created in Spectrum Management Console.

Usage

`dbdriver delete --n` *DriverName*

Required	Argument	Description
Yes	`--n` *DriverName*	Specifies the name of the JDBC driver you want to delete. To view a list of drivers, use the `dbdriver list` command.

Example

This example deletes a JDBC driver named MyDriver.

```
dbconnection delete --n MyDriver
```

dbdriver export

The `dbdriver export` command exports a JDBC driver definition to a JSON file. It does not export driver files, only the driver definition created in Spectrum Management Console.

Usage

`dbdriver export --n` *DriverName* `--o` *OutputDirectory*

Required	Argument	Description
Yes	`--n` *DriverName*	Specifies the name of the database driver you want to export. If the driver name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the driver name you can use the `dbdriver list` command to get a list of the driver names.
No	`--o` *OutputDirectory*	Specifies the directory to which you want to export the database driver. The path you specify here is relative to the directory where you are running the Administration Utility. If you omit this argument, the database driver is exported to the directory containing the Administration Utility.

Example

This example exports the database driver named "My Driver" to a folder named `exported` which is a subfolder in the location where you have installed the Administration Utility.

```
dbdriver export --n "My Driver" --o exported
```

dbdriver import

The `dbdriver import` command imports a JDBC database driver definition file into the server. Database driver definition files are created by exporting a database driver definition from the server using the `dbdriver export` command. You can only import database driver definitions that were exported from the same version of Spectrum Technology Platform.

Usage

`dbdriver import --f` *DriverDefinitionFile* `--u` *TrueOrFalse*

Required	Argument	Description
Yes	`--f` *DriverDefinitionFile*	Specifies the database driver JSON file you want to import. Directory paths you specify here are relative to the location where you are running the Administration Utility.
No	`--u` *TrueOrFalse*	Specifies whether to overwrite the existing database driver definition if a database driver with the same name is already on the server, where *TrueOfFalse* is one of the following: **true** If there is a database driver on the server with the same name as the one you are importing, the driver on the server will be overwritten. This is the default setting. **false** If there is a database driver on the server with the same name as the one you are importing, the driver will not be imported.

Example

This example imports the database driver definition named `MyDatabaseDriver.json` which is located in a subfolder named `exported` in the location where you are running the Administration Utility.

```
dbdriver import --f exported\MyDatabaseDriver.json
```

dbdriver list

The `dbdriver list` command returns a list of the database drivers defined on the Spectrum Technology Platform server.

Usage

```
dbdriver list
```

Dataflows

dataflow delete

The `dataflow delete` command removes a dataflow from your system.

Usage

`dataflow delete --d` *DataflowName*

Required	Argument	Description
Yes	`--d` *DataflowName*	Specifies the dataflow to delete. If the dataflow name contains spaces, enclose the dataflow name in quotes.

Example

This example deletes the dataflow named My Dataflow.

```
dataflow delete --d "My Dataflow"
```

dataflow export

The `dataflow export` command exports a dataflow from the server to a `.df` file. The dataflow can then be imported to another server.

Note: Dataflows can only be exchanged between identical versions of Spectrum Technology Platform.

Usage

`dataflow export --d` *DataflowName* `--e` *TrueOrFalse* `--o` *OutputDirectory*

Required	Argument	Description
Yes	`--d` *DataflowName*	Specifies the name of the dataflow you want to export. If the dataflow name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the exact dataflow name you can use the `dataflow list` command to get a list of the dataflow names.
Yes	`--e` *TrueOrFalse*	Specifies whether to export the exposed version of the dataflow, where *TrueOrFalse* is one of the following: **true** Export the exposed version of the dataflow. **false** Export the most recently saved version of the dataflow.
No	`--o` *OutputDirectory*	Specifies the directory to which you want to export the dataflow. The path you specify here is relative to the directory where you are running the Administration Utility. If you omit this argument, the dataflow is exported to the directory containing the Administration Utility.

Example

This example exports the exposed version of a dataflow named "My Dataflow" to a folder named `exported` which is a subfolder in the location where you have installed the Administration Utility.

```
dataflow export --d "My Dataflow" --e true --o exported
```

dataflow expose

The `dataflow expose` command makes the dataflow available for execution. For service dataflows, exposing the dataflow makes the service available to web service requests and API calls, and makes it available for setting logging levels. For subflows, exposing the dataflow makes the subflow available for use in a dataflow. For job dataflows, exposing the dataflow makes it possible to run the job through the Job Executor command line tool. To expose a process flow use the `processflow expose` command.

Note: If you use dataflow visioning in Spectrum Enterprise Designer, the `dataflow expose` command exposes the most recently saved version of the dataflow.

Usage

`dataflow expose --d` *DataflowName*

Required	Argument	Description
Yes	`--d` *DataflowName*	Specifies the name of the dataflow you want to expose. If the dataflow name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the exact dataflow name you can use the `dataflow list` command to get a list of the dataflow names.

Example

This example exposes the dataflow named "My Dataflow".

```
dataflow expose --d "My Dataflow"
```

dataflow import

The `dataflow import` command imports a dataflow file (a `.df` file) into the server. Dataflow files are created by exporting a dataflow from the server using the `dataflow export` command.

Usage

`dataflow import --f` *DataflowFile* `--u` *TrueOrFalse* `--p` *Path* `--c` *TrueOrFalse*

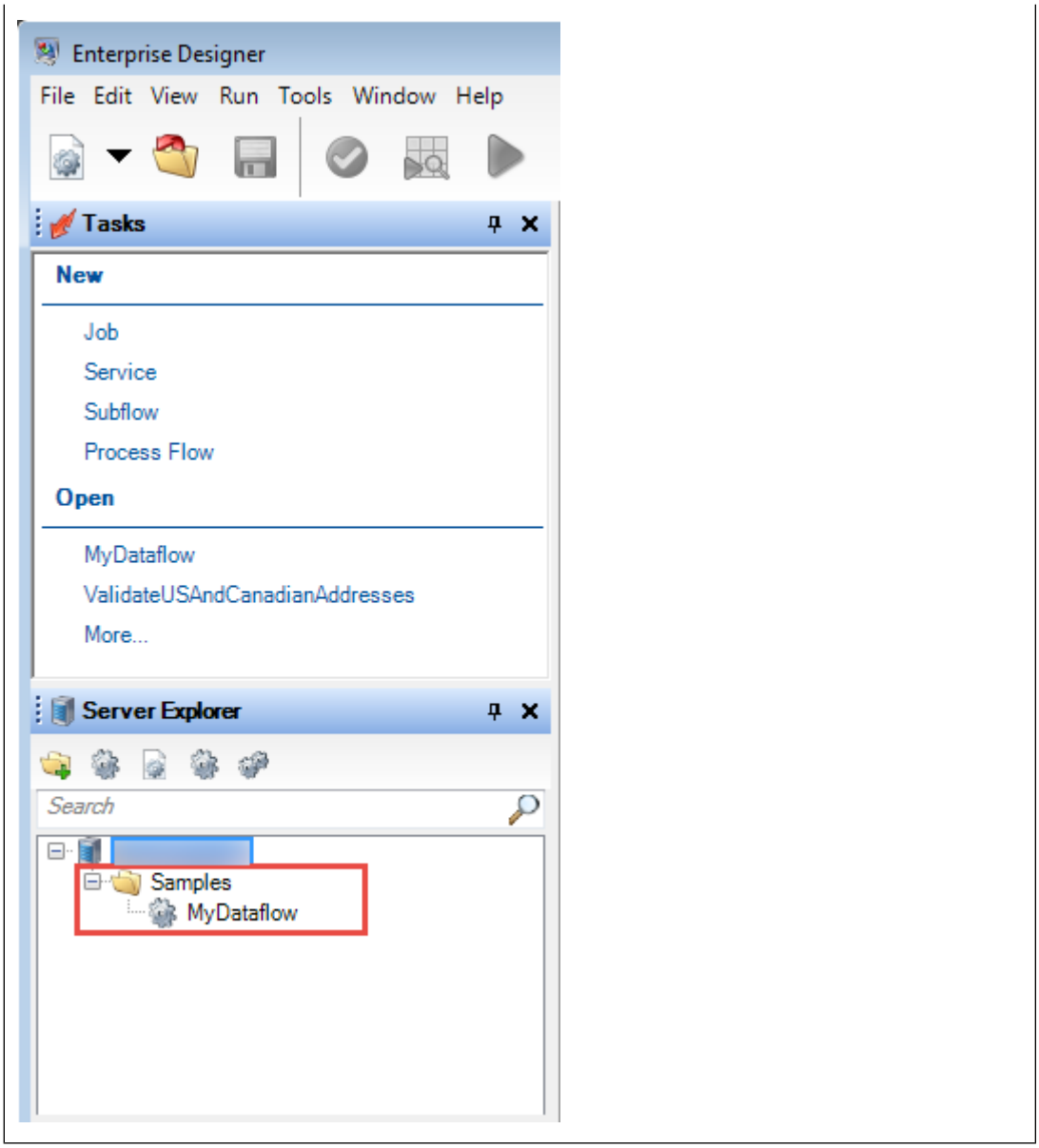
Required	Argument	Description
Yes	`--f` *DataflowFile*	Specifies the dataflow file (the `.df` file) you want to import. Relative directory paths are relative to the location where you are running the Administration Utility. You can also specify an absolute path.
No	`--u` *TrueOrFalse*	Specifies whether to overwrite the existing dataflow if a dataflow with the same name is already on the server, where *TrueOfFalse* is one of the following: **true** If there is a dataflow on the server with the same name as the dataflow you are importing, the dataflow on the server will be overwritten. This is the default setting. **false** If there is a dataflow on the server with the same name as the dataflow you are importing, the dataflow will not be imported.

Required	Argument	Description
No	`--p` *Path*	Specifies the Spectrum Enterprise Designer Server Explorer folder to import the flow into.
No	`--c` *TrueOrFalse*	Specifies whether to create the folder specified in `--p` if it does not exist. **true** Create the folder specified in `--p` if it does not exist. Default. **false** Do not create the folder specified in `--p` if it does not exist. The flow will not be imported unless the folder specified in `--p` exists.

Example

This example imports the dataflow named `MyDataflow.df` which is located in a subfolder named `exported` in the location where you are running the Administration Utility. The dataflow will be imported into the `Samples` folder in Spectrum Enterprise Designer.

```
dataflow import --f exported\MyDataflow.df --p Samples
```



dataflow list

The `dataflow list` command lists all the dataflows on the server. For each dataflow, certain information is displayed: the dataflow name, type of dataflow (job, service, or subflow), and whether the dataflow is exposed.

Usage

```
dataflow list
```

dataflow lock list

The `dataflow lock list` command lists the dataflows that are locked for editing by a user. Dataflows are locked when a user opens the dataflow in Spectrum Enterprise Designer, and unlocked when the user closes the dataflow in Spectrum Enterprise Designer.

Usage

```
dataflow lock list
```

dataflow sourcesink list

The `dataflow sourcesink list` command lists the stages in a dataflow that specify the input for the dataflow and the stages that specify the output from the dataflow.

Usage

`dataflow sourcesink list --d` *DataflowName* `--e` *TrueOrFalse* `--o` *TrueOrFalse*

Required	Argument	Description
Yes	`--d` *DataflowName*	Specifies the name of the dataflow whose sources and sinks you want to list. If the dataflow name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the exact dataflow name you can use the `dataflow list` command to get a list of the dataflow names.
Yes	`--e` *TrueOrFalse*	Specifies whether to list the sources and sinks for the exposed version of the dataflow or the latest saved version. **true** List the sources and sinks in the exposed version of the dataflow. **false** List the sources and sinks in the most recently saved version of the dataflow.
No	`--o` *TrueOrFalse*	Specifies whether to list only those sources and sinks that allow file overrides at runtime. A file override is when you specify at runtime a different file for the stage to read from or write to, overriding the file specified in the stage itself. Stages that support file overrides include Read from File and Write to File. Stages that do not support file overrides include Write to Null and Terminator.

Required	Argument	Description
		true List only those stages that support file overrides. **false** List all sources and sinks. This is the default setting.

Example

This example lists the sources and sinks in the exposed version of a dataflow named "My Dataflow". All sources and sinks are listed, even those that do not allow file overrides.

```
dataflow sourcesink list --d "My Dataflow" --e true
```

dataflow unexpose

The `dataflow unexpose` command makes a dataflow unavailable for execution as either a service or as a job.

Usage

`dataflow unexpose --d` *DataflowName*

Required	Argument	Description
Yes	`--d` *DataflowName*	Specifies the name of the dataflow you want to unexpose. If the dataflow name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the exact dataflow name you can use the `dataflow list` command to get a list of the dataflow names.

Example

This example unexposes the dataflow named "My Dataflow".

```
dataflow unexpose --d "My Dataflow"
```

dataflow unlock

The `dataflow unlock` command unlocks a dataflow, making it possible for other users to edit it in Spectrum Enterprise Designer. In normal use, dataflows are unlocked automatically when a user closes the dataflow in Spectrum Enterprise Designer. In certain situations, it may be necessary for an administrator to unlock a dataflow using the `dataflow unlock` command. For example, if a user opens a dataflow in Spectrum Enterprise Designer and leaves for the day, the dataflow remains locked, preventing other users from editing it. In this case, you could use the `dataflow unlock` command to unlock the dataflow. Once a dataflow is unlocked, Spectrum Enterprise Designer users must close and reopen the flow in order to be able to save it.

In order to use the `dataflow unlock` command you must have the **Dataflows - Unlock** permission.

Warning: Unlocking a dataflow will prevent the user who had locked the dataflow from saving any unsaved changes.

Usage

`dataflow unlock --d` *DataflowName*

Required	Argument	Description
Yes	`--d` *DataflowName*	Specifies the dataflow to unlock. If the dataflow name contains spaces, enclose the dataflow name in quotes.

dataflow version list

The `dataflow version list` command lists all available versions of a specific dataflow. Specify the dataflow name using the `--n` command parameter. When you create dataflows, Spectrum maintains the dataflows until you delete them, and applies a save version to each one (1.0.0, 1.0.1, etc.) .

Usage

`dataflow version list --n` *DataflowName*

Required	Argument	Description
Yes	`--n` *DataflowName*	Specifies the name of the dataflow whose versions you want to list. If the dataflow name contains spaces, enclose the name in double quotes. **Tip:** If you are unsure of the exact dataflow name you can use the `dataflow list` command to get a list of the dataflow names.

Entities

Export entity security overrides

Export security overrides for a role or user, to JSON format, using the Administration Utility Command Line Interface (CLI).

Usage

`entity override export --e` *role_or_user_value* `--p` *role_or_user_literal*

Required	Argument	Description
Yes	`--e` *role_or_user_value*	Specifies the value (userName/RoleName) of the role or user for which the overrides are exported.
Yes	`--p` *role_user_literal*	Specifies the literal name of the "Role" or "User".

Examples

To export overrides for a user named "Sally", use this syntax:

```
entity override export --e user --p Sally
```

To export overrides for "Sally" who has the role "designer," use this syntax:

```
entity override export --e Sally --p designer
```

Folders

folder browse

The `folder browse` command lists the contents of a Server Explorer folder.

Usage

`folder browse --p` *Path*

Required	Argument	Description
No	`--p` *Path*	Specifies the folder whose contents you want to list. If you omit this parameter the contents of the root folder are listed.

folder create

The `folder create` command creates a folder in Server Explorer.

Usage

`folder create --p` *Path*

Required	Argument	Description
Yes	`--p` *Path*	Specifies the path of the folder you want to create.

Example

This example creates a folder named `Example123` inside the folder `ExampleABC`.

```
folder create --p ExampleABC/Example123
```

folder delete

The `folder delete` command deletes a folder from Server Explorer.

Usage

`folder delete --p` *Path* `--r` *TrueFalse*

Required	Argument	Description
Yes	`--p` *Path*	Specifies the path of the folder you want to delete.
No	`--r` *TrueFalse*	Specifies whether to delete the folder if it contains a flow or subfolders. **true** Delete the folder if it contains flows or subfolders. **false** Do not delete the folder if it contains flows or subfolders. Default.

Example

This example deletes a folder named `Example123`. The folder will be deleted even if it contains flows or subfolders.

```
folder delete --p ExampleABC/Example123 --r true
```

folder move

The `folder move` command moves a folder in Server Explorer to another location.

Usage

`folder move --p` *Path* `--t` *Target*

Required	Argument	Description
Yes	`--p` *Path*	Specifies the path of the folder you want to move.
Yes	`--t` *Target*	Specifies the path to which you want to move the folder.

Example

This example moves a folder named `ExampleABC` into the folder `Example123`.

```
folder move --p ExampleABC --t Example123
```

folder rename

The `folder rename` command changes the name of a folder in Server Explorer.

Usage

`folder rename --p` *Path* `--n` *NewName*

Required	Argument	Description
Yes	`--p` *Path*	Specifies the path of the folder you want to rename.
Yes	`--n` *NewName*	Specifies the new name of the folder.

Information Extraction

iemodel delete

The `iemodel delete` command returns a list of all Spectrum Information Extraction models.

Usage

`iemodel delete --n` *modelName*

Required	Argument	Description
Yes	`--n` *modelName*	Specifies the name of the model you want to delete. Directory paths you specify here are relative to the location where you are running the Administration Utility.

Example

This example deletes the model called "MyModel".

```
iemodel delete --n MyModel
```

iemodel evaluate model

The `iemodel evaluate` command evaluates a Spectrum Information Extraction model that has previously been trained.

Usage

`iemodel evaluate model --n` *modelName* `--t` *testFileName* `--o` *outputFileName* `--c` *categoryCount* `--d` *trueOrfalse*

Required	Argument	Description
Yes	`--n` *modelName*	Specifies the name and location of the model you want to evaluate. Directory paths you specify here are relative to the location where you are running the Administration Utility.
Yes	`--t` *testFileName*	Specifies the name and location of the test file used to evaluate the model.
No	`--o` *outputFileName*	Specifies the name and location of the output file that will store the evaluation results.
No	`--c` *categoryCount*	Specifies the number of categories in the model; must be a numeric value. **Note:** It is applicable only for Text Classification model.
No	`--d` *trueOrfalse*	Specifies whether to display a table with entity wise detailed analysis; the value must be `true` or `false`, as below: **true** Detailed evaluation results are required. **false** Detailed evaluation results are not required. The default is `false`. The *Model Evaluation Results* table, and *Confusion Matrix* with its columns, as described below, display the counts for each entity. **Note:** If the command is run without this argument or with the argument value `false`, the *Model Evaluation Results*

Required	Argument	Description
		table and *Confusion Matrix* are not displayed. Only the *Model Evaluation Statistics* are displayed.

Output

Model Evaluation Statistics

Executing this command displays these evaluation statistics in a tabular format:

- **Precision**: It is a measure of exactness. Precision defines the proportion of correctly identified tuples.
- **Recall**: It is a measure of completeness of the results. Recall can be defined as a fraction of relevant instances that are retrieved.
- **F1 Measure**: It is the measure of the accuracy of a test. The computation of F1 score takes into account both precision and recall of the test. It can be interpreted as the weighted average of the precision and recall, where F1 score reaches its best value at 1 and worst at 0.
- **Accuracy**: It measures the degree of correctness of results. It defines the closeness of the measured value to the known value.

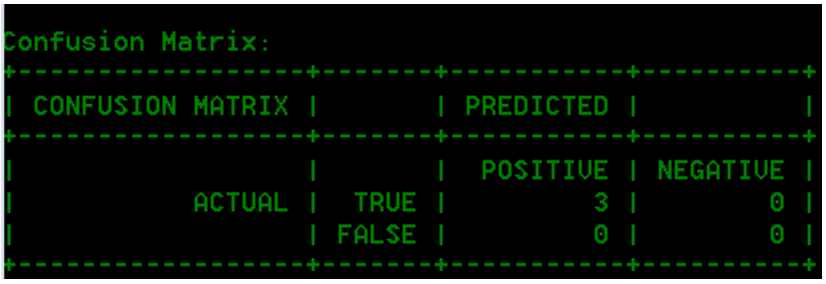
Model Evaluation Results

If the command is run with the argument `--d true`, the match counts of all the entities are displayed in a tabular format. The columns of the table are:

Input Count	The number of occurrences of the entity in the input data.
Mismatch Count	The number of times the entity match failed.
Match Count	The number of times the entity match succeeded.

Confusion Matrix

The *Confusion Matrix* (shown below) allows visualization of how an algorithm performs. It illustrates the performance of a classification model.



```
Confusion Matrix:
+------------------+-------+-----------+-----------+
| CONFUSION MATRIX |       | PREDICTED |           |
+------------------+-------+-----------+-----------+
|                  |       |  POSITIVE | NEGATIVE  |
|           ACTUAL |  TRUE |         3 |         0 |
|                  | FALSE |         0 |         0 |
+------------------+-------+-----------+-----------+
```

The column represents the instances in a predicted class while the row represents the instances in an actual class. Some of the terms associated with the confusion matrix are:

Actual	The number of occurrences of the entity in the actual class.
Predicted	The number of occurrences of the entity in the predicted class.

TP	**True Positive:** The number of entity occurrences predicted as positive and actually true as well.
TN	**True Negative:**The number of entity occurrences predicted as negative but actually true.
FP	**False Positive:** The number of entity occurrences predicted as positive but actually false.
FN	**False Negative:** The number of entity occurrences predicted as negative and actually false as well.

Example

This example:

- Evaluates the model called "MyModel"
- Uses a test file called "ModelTestFile" in the same location
- Stores the output of the evaluation in a file called "MyModelTestOutput"
- Specifies a category count of 4
- Specifies that a detailed analysis of the evaluation is required

```
iemodel evaluate model --n MyModel --t
C:\Spectrum\IEModels\ModelTestFile --o
C:\Spectrum\IEModels\MyModelTestOutput --c 4 --d true
```

iemodel evaluate train_model

The `iemodel evaluate train_model` command evaluates and trains an existing **Spectrum Information Extraction** model. This function cannot be performed on a new model.

> **Note:** For better results on evaluation and training of an existing **Spectrum Information Extraction**, use this command: `iemodel trainAndevaluate model`. For details, see **iemodel trainAndevaluate model** on page 336.

Usage

`iemodel evaluate train_model --f` *trainingOptionsFile* `--u` *trueOrFalse* `--o` *outputFileName* `--c` *categoryCount* `--d` *trueOrfalse*

Required	Argument	Description
Yes	`--f` *trainingOptionsFile*	Specifies the name and location of the training options file used to train the model. Directory paths you specify here are relative to the location where you are running the Administration Utility.

Required	Argument	Description
No	`--u` *overWriteIfExists*	Specifies whether to overwrite the existing trained model (if one exists). *TrueOrFalse* is one of the following: **true** Overwrites the existing model. **false** Does not overwrite the existing model.
No	`--o` *outputFileName*	Specifies the name and location of the output file that will store the evaluation results.
No	`--c` *categoryCount*	Specifies the number of categories in the model; must be a numeric value. **Note:** It is applicable only for Text Classification model.
No	`--d` *trueOrfalse*	Specifies whether to display a table with entity wise detailed analysis; the value must be `true` or `false`, as below: **true** Detailed evaluation results are required. **false** Detailed evaluation results are not required. The default is `false`. The *Model Evaluation Results* table, with its columns as described below,displays the counts per entity. **Note:** If the command is run without this argument or with the argument value `false`, the Model Evaluation Results table is not displayed. Only the Model Evaluation Statistics are displayed.

Output

Model Evaluation Statistics

Executing this command displays these evaluation statistics in a tabular format:

- Precision
- Recall
- F1 Measure

Model Evaluation Results

If the command is run with the argument `--d true`, the match counts of all the entities are displayed in a tabular format. The columns of the table are:

Input Count	The number of occurrences of the entity in the input data.

Mismatch Count	The number of times the entity match failed.
Match Count	The number of times the entity match succeeded.

Example

This example:

- Uses a training options file called "ModelTrainingFile" that is located in "C:\Spectrum\IEModels"
- Overwrites any existing output file of the same name
- Stores the output of the evaluation in a file called "MyModelTestOutput"
- Specifies a category count of 4
- Specifies that a detailed analysis of the evaluation is required

```
iemodel evaluate train_model --f
C:\Spectrum\IEModels\ModelTrainingFile --u true --o
C:\Spectrum\IEModels\MyModelTestOutput --c 4 --d true
```

iemodel export

The `iemodel export` command exports an Spectrum Information Extraction model and its metadata.

Usage

`iemodel export --n` *modelName* `--o` *outputDirectory*

Required	Argument	Description
Yes	`--n` *modelName*	Specifies the name of the model you want to export. Directory paths you specify here are relative to the location where you are running the Administration Utility.
Yes	`--o` *outputDirectory*	Specifies the location of the folder that will store the exported model and its metadata.

Example

This example exports a model named `MyModel` that places the output in a folder called "MyModelExport", which is located in "C:\Spectrum\IEModels\MyModelExport".

```
iemodel export --n MyModel --o
C:\Spectrum\IEModels\MyModelExport
```

iemodel import

The `iemodel import` command imports an Spectrum Information Extraction model and its metadata.

Usage

`iemodel import --n` *modelName* `--o` *inputDirectory* `--u` *trueOrFalse*

Required	Argument	Description
Yes	`--n` *modelName*	Specifies the name of the model you want to import. Directory paths you specify here are relative to the location where you are running the Administration Utility.
Yes	`--o` *inputDirectory*	Specifies the location of the folder that will store the imported model and its metadata.
No	`--u` *overWriteIfExists*	Specifies whether to overwrite the existing model (if one exists). *TrueOrFalse* is one of the following: **true** Overwrites the existing model. **false** Does not overwrite the existing model.

Example

This example imports a model named `MyModel` that stores the model in a folder called "MyModelExport", which is located in "C:\Spectrum\IEModels\MyModelExport". It also overwrites any existing model of the same name.

```
iemodel import --n MyModel --o
C:\Spectrum\IEModels\MyModelExport --u true
```

iemodel list

The `iemodel list` command returns a list of all Spectrum Information Extraction models.

Usage

```
iemodel list
```

Example

This example lists all models.

```
iemodel list
```

iemodel train

The `iemodel train` command trains an Spectrum Information Extraction model. It calls your training options file, which points to your input file and applies the options you have specified.

Usage

`iemodel train --f` *trainingOptionsFile* `--u` *trueOrFalse*

Required	Argument	Description
Yes	`--f` *trainingOptionsFile*	Specifies the name and location of the training options file used to train the model. Directory paths you specify here are relative to the location where you are running the Administration Utility.
No	`--u` *trueOrFalse*	Specifies whether to overwrite the existing model with the same name (if one exists), where *TrueOrFalse* is one of the following: **true** Overwrites the existing model. **false** Does not overwrite the existing model.

Example

This example trains a model listed in the *TrainingOptions.xml* file that is stored the `C:` drive and overwrites any existing model of the same name.

```
iemodel train --f c:/TrainingOptions.xml --u true
```

iemodel trainAndevaluate model

The `iemodel trainAndevaluate model` command evaluates and trains a new model as well as an existing model. In case of an existing model you need to overwrite it with the newly trained model by using "true" for the argument --u in the command.

This command calls your training options file and provides an optional output file with evaluation results, should you choose to produce that file.

Usage

`iemodel trainAndevaluate model --f` *trainingOptionsFile* `--u` *trueOrFalse* `--o` *outputFileName* `--c` *categoryCount* `--d` *trueOrfalse*

Required	Argument	Description
Yes	`--f` *trainingOptionsFile*	Specifies the name and location of the training options file used to train the model. Directory paths you specify here are relative to the location where you are running the Administration Utility.
No	`--u` *overWriteIfExists*	Specifies whether to overwrite the existing trained model (if one exists). **true** Overwrites the existing model. **false** Does not overwrite the existing model.
No	`--o` *outputFileName*	Specifies the name and location of the output file that will store the evaluation results.
No	`--c` *categoryCount*	Specifies the number of categories in the model; must be a numeric value. **Note:** It is applicable only for Text Classification model.
No	`--d` *trueOrfalse*	Specifies whether to display a table with entity wise detailed analysis; the value must be `true` or `false`, as below: **true** Detailed evaluation results are required. **false** Detailed evaluation results are not required. The default is `false`. The *Model Evaluation Results* table, and *Confusion Matrix* with its columns, as described below, display the counts for each entity. **Note:** If the command is run without this argument or with the argument value `false`, the *Model Evaluation Results* table and *Confusion Matrix* are not displayed. Only the *Model Evaluation Statistics* are displayed.

Output

Model Evaluation Statistics

Executing this command displays these evaluation statistics in a tabular format:

- **Precision**: It is a measure of exactness. Precision defines the proportion of correctly identified tuples.
- **Recall**: It is a measure of completeness of the results. Recall can be defined as a fraction of relevant instances that are retrieved.
- **F1 Measure**: It is the measure of the accuracy of a test. The computation of F1 score takes into account both precision and recall of the test. It can be interpreted

as the weighted average of the precision and recall, where F1 score reaches its best value at 1 and worst at 0.

- **Accuracy**: It measures the degree of correctness of results. It defines the closeness of the measured value to the known value.

Model Evaluation Results

If the command is run with the argument `--d true`, the match counts of all the entities are displayed in a tabular format. The columns of the table are:

Input Count	The number of occurrences of the entity in the input data.
Mismatch Count	The number of times the entity match failed.
Match Count	The number of times the entity match succeeded.

Confusion Matrix

The *Confusion Matrix* (shown below) allows visualization of how an algorithm performs. It illustrates the performance of a classification model.

```
Confusion Matrix:
+------------------+-------+-----------+-----------+
| CONFUSION MATRIX |       | PREDICTED |           |
+------------------+-------+-----------+-----------+
|                  |       |  POSITIVE | NEGATIVE  |
|           ACTUAL |  TRUE |         3 |         0 |
|                  | FALSE |         0 |         0 |
+------------------+-------+-----------+-----------+
```

The column represents the instances in a predicted class while the row represents the instances in an actual class. Some of the terms associated with the confusion matrix are:

Actual	The number of occurrences of the entity in the actual class.
Predicted	The number of occurrences of the entity in the predicted class.
TP	**True Positive:** The number of entity occurrences predicted as positive and actually true as well.
TN	**True Negative:**The number of entity occurrences predicted as negative but actually true.
FP	**False Positive:** The number of entity occurrences predicted as positive but actually false.
FN	**False Negative:** The number of entity occurrences predicted as negative and actually false as well.

Example

This example:

- Uses a training options file called "ModelTrainingFile" that is located in "C:\Spectrum\IEModels"
- Overwrites any existing output file of the same name

- Stores the output of the evaluation in a file called "MyModelTestOutput"
- Specifies a category count of 4
- Specifies that a detailed analysis of the evaluation is required

```
iemodel trainAndevaluate model --f
C:\Spectrum\IEModels\ModelTrainingFile --u true --o
C:\Spectrum\IEModels\MyModelTestOutput --c 4 --d true
```

Jobs

job history list

The `job history list` command shows the execution history for a job.

Usage

`job history list --j` *JobName* `--f` *FromDateTime* `--t` *ToDateTime*

Required	Argument	Description
Yes	`--j` *JobName*	Specifies the name of the job whose history you want to get. If the job name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the exact dataflow name you can use the `dataflow list` command to get a list of the dataflow names.
No	`--f` *FromDateTime*	If you want to see the history for a specific date and time range, specify the starting date and time for the range, in the format MM-dd-yyyy HH:mm:ss. For example, December 31, 2014 1:00 PM would be specified as `12-31-2014 13:00:00`. When you specify a date and time range, the history list will include jobs that started execution on or after the date you specified in the `--f` argument and before the date you specify in the `--t` argument. If you omit this argument the history will include jobs that started execution on the current date.
No	`--t` *ToDateTime*	If you want to see the history for a specific date and time range, specify the ending date and time for the range, in the format MM-dd-yyyy HH:mm:ss. For example, December 31, 2014 1:00 PM would be specified as `12-31-2014 13:00:00`.

Required	Argument	Description
		When you specify a date and time range, the history list will include jobs that started execution on or after the date you specified in the `--f` argument and before the date you specify in the `--t` argument. If you omit this argument the history will include all jobs that started execution on or after the date specified in the `--f` argument.

Example

This example gets the status of the job named "My Job".

```
job history list --j "My Job"
```

job execute

The `job execute` command runs one or more jobs. After the job runs, the job name and job ID are returned in the format:

<JobName=JobID>

Usage

`job execute --j` *JobNames* `--f` *JobPropertyFile* `--i` *PollInterval* `--d` *ReportDelimiter* `--n` *NotificationEmails* `--o` *OptionPropertyFile* `--r` *ReportTrueOrFalse* `--t` *Timeout* `--w` *WaitTrueOrFalse* `--l` *FileOverrides* `--v` *VerboseTrueOrFalse*

Required	Argument	Description
Yes	`--j` *JobNames*	Specifies the name of one or more jobs to run. If you specify more than one job, separate each job with a comma. Jobs run in the order you list them. If the job name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the exact dataflow name you can use the `dataflow list` command to get a list of the dataflow names.
No	`--f` *JobPropertyFile*	Specifies the path to a job property file. A job property file contains arguments that control the execution of jobs. For more information, see **Using a Job Property File** on page 200.
No	`--i` *PollInterval*	When `--w` is set to `true`, use this option to specify how often to check for completed jobs, in seconds. The default is 5.

Required	Argument	Description
No	`--d` *ReportDelimiter*	Sets the delimiter character to use in the report output when you also specify `--w true` or `--r true`. The default is the pipe character (|).
No	`--n` *NotificationEmails*	Specifies one or more email addresses to receive notifications about the status of jobs, as configured in Spectrum Management Console. Separate each email address with a comma.

No	`--o` *OptionPropertyFile*	Specifies a path to a flow options property file. Use a flow options property file to set options for stages in the flow. In order to set flow options using a property file, you must configure the flow to expose stage options at runtime. For more information, see **Adding Flow Runtime Options** on page 210.
		
		For example, a flow options properties file for a flow that contains an Assign GeoTAX Info stage may look like this:
		
		`OutputCasing=U`
		`UseStreetLevelMatching=N`
		`TaxKey=T`
		`Database.GTX=gsl`

No	`--r` *ReportTrueOrFalse*	Specify `true` to return a detailed report about the job. This option only works if you also specify `--w true` . The report contains the following information: • **Position 1** - Name of job • **Position 2** - Job process ID • **Position 3** - Status • **Position 4** - Start Date-Time (MM/DD/YYYY HH:MM:SS) • **Position 5** - End Date-Time (MM/DD/YYYY HH:MM:SS) • **Position 6** - Number of successful records • **Position 7** - Number of failed records • **Position 8** - Number of malformed records • **Position 9** - Currently unused For example, `MySimpleJob|4|succeeded|04/09/2010 14:50:47|04/09/2010 14:50:47|100|0|0|` The information is delimited using the delimiter specified in the `--d` argument.
No	`--t` *Timeout*	Sets the timeout value for synchronous mode, in seconds. The default is 3600.

Required	Argument	Description
No	`--w` *WaitTrueOrFalse*	Specify `true` to run jobs one at a time in synchronous mode. Specify `false` to run all the jobs at the same time. The default is false.
No	`--l` *FileOverrides*	Overrides the input and output file and file format. Separate file and format specifications with a comma. For more information see **Overriding Job Files** on page 344 and **Overriding File Format** on page 346.
No	`--v` *VerboseTrueOrFalse*	Specify `true` to return information about the arguments used to run the job and other details about the job execution.

Example

This example runs a job named Example1. It returns a comma-delimited report. Note that `--w true` is specified because this is required to return a report even if only one job is running. The input file specified in the Read from File stage is changed from what is specified in the stage to a different file named `CandidateHomes2.csv`. Verbose output is also returned.

```
job execute --j Example1 --w true --d "," --r true --l "Read
from
File=file:/e:/SampleDataflows/DataFiles/DataFiles/CandidateHomes2.csv"
--v true
```

Using a Job Property File

A job property file contains arguments that control the execution of jobs when you use the job executor or the Administration Utility to run a job. Use a job property file if you want to reuse arguments by specifying a single argument at the command line (`-f`) rather than specifying each argument individually at the command line.

To create a property file, create a text file with one argument on each line.

```
d %
h spectrum.mydomain.com
i 30
j validateAddressJob1
u user
p password
s 8888
t 9999
w true
```

The job property file can contain these arguments:

Required	Argument	Description
No	`?`	Prints usage information.
No	`d` *delimiter*	Sets instance/status delimiter. This appears in synchronous output only.
No	`e`	Use a secure HTTPS connection for communication with the Spectrum Technology Platform server.
No	`h` *hostname*	Specifies the name or IP address of the Spectrum Technology Platform server.
No	`i` *pollinterval*	Specifies how often to check for completed jobs, in seconds. This applies only in synchronous mode.
Yes	`j` *jobname*	A comma-separated list of jobs to run. Job names are case-sensitive. Jobs are started in the order listed.
No	`n` *emaillist*	Specifies a comma-separated list of additional email addresses for configured job notifications.
Yes	`p` *password*	The password of the user.
No	`r`	Returns a delimited list with this information about the job written to standard output: • **Position 1** - Name of job • **Position 2** - Job process ID • **Position 3** - Status • **Position 4** - Start Date - Time (MM/DD/YYYY HH:MM:SS) • **Position 5** - End Date - Time (MM/DD/YYYY HH:MM:SS) • **Position 6** - Number of successful records • **Position 7** - Number of failed records • **Position 8** - Number of malformed records • **Position 9** - Currently unused The information is delimited using the delimiter specified in the -d argument. For example: `MySimpleJob|4|succeeded|04/09/2019 14:50:47|04/09/2019 14:50:47|100|0|0|`
No	`s` *port*	The socket (port) on which the Spectrum Technology Platform server is running. The default is 8080.
No	`t` *timeout*	Sets the timeout (in seconds) for synchronous mode. The default is 3600. The maximum is 2147483. This is a global, aggregate timeout and represents the maximum time to wait for all spawned jobs to complete.
Yes	`u` *username*	The login name of the user.

Required	Argument	Description
No	`v`	Return verbose output.
No	`w`	Specifies to wait for jobs to complete in a synchronous mode.

Using Both Command Line Arguments and a Property File

A combination of both command-line entry and property file entry is also valid. For example:

```
java -jar jobexecutor.jar -f /dcg/job.properties -j job1
```

In this case command line arguments take precedence over arguments specified in the properties file. In the above example, the job job1 would take precedence over a job specified in the properties file.

Overriding Job Files

When you run a job at the command line using job executor or the Administration Utility, you can override the input file specified in the flow's source stage (such as Read from File), as well as the output file specified in the flow's sink stage (such as Write to File).

To do this in job executor, specify this command at the end of the job executor command:

`StageName=Protocol:FileName`

In the Administration Utility, use the `--l` argument in the `job execute` command:

`--l` *`StageName=Protocol:FileName`*

Where:

StageName

The stage label shown under the stage's icon in the flow in Spectrum Enterprise Designer. For example, if the stage is labeled "Read from File" you would specify `Read from File` for the stage name.

To specify a stage within an embedded flow or a subflow, preface the stage name with the name of the embedded flow or subflow, followed by a period then the stage name:

`EmbeddedOrSubflowName.StageName`

For example, to specify a stage named Write to File in a subflow named Subflow1, you would specify:

```
Subflow1.Write to File
```

To specify a stage in an embedded flow that is within another embedded flow, add the parent flow, separating each with a period. For example, if Embedded Dataflow 2 is inside Embedded Dataflow 1, and you want to specify the Write to File stage in Embedded Dataflow 2, you would specify this:

```
Embedded Dataflow 1.Embedded Dataflow 2.Write to File
```

Protocol

A communication protocol that can be one of these types:

file Use the file protocol if the file is on the same machine as the Spectrum Technology Platform server. For example, on Windows specify:

```
"file:/C:/myfile.txt"
```

On Linux specify:

```
"file:/testfiles/myfile.txt"
```

esclient Use the esclient protocol if the file is on the computer where you are executing the job if it is a different computer from the one running the Spectrum Technology Platform server. Use this format:

esclient:*ComputerName/path to file*

For example,

```
esclient:mycomputer/testfiles/myfile.txt
```

> **Note:** If you are executing the job on the server itself, you can use either the file or esclient protocol, but are likely to have better performance using the file protocol.

If the host name of the Spectrum Technology Platform server cannot be resolved, you may get the error "Error occurred accessing file". To resolve this issue, open this file on the server: *SpectrumDirectory*`/server/conf/spectrum-container.properties`. Set the `spectrum.runtime.hostname` property to the IP address of the server.

esfile Use the esfile protocol if the file is on a file server. The file server must be defined in Spectrum Management Console as a resource. Use this format:

esfile://*file server/path to file*

For example,

```
esfile://myserver/testfiles/myfile.txt
```

Where myserver is an FTP file server resource defined in Spectrum Management Console.

webhdfs Use the webhdfs protocol if the file is on a Hadoop Distributed File Server. The HDFS server must be defined in Spectrum Management Console as a resource. Use this format:

webhdfs://*file server/path to file*

For example,

```
webhdfs://myserver/testfiles/myfile.txt
```

Where myserver is an HDFS file server resource defined in Spectrum Management Console.

FileName

The full path to the file you want to use as input or output.

Note: You must use forward slashes in file paths. Do not use backslashes.

To specify multiple overrides, separate each override with a comma.

Example File Override

This example executes a job named TestJob. Instead of writing the output to the file specified in the Write to File stage, it will write the output to `outputoverride.txt`.

```
job execute --j TestJob --l "Write to
File=file:/Users/me/outputoverride.txt"
```

Example Override Both Read and Write File Formats

To specify both `Read from` and `Write to` files, separate the two locations with a comma.

```
job execute --j J1 --v true --w true --l "Read from
File=file://C:/tmp/input3.csv,Write to
File=file://C:/tmp/my_new_output.csv"
```

Overriding File Format

When you run a job using job executor or the Administration Utility, you can override the file layout (or schema) of the file specified in the flow's Read from File stage and Write to File stage.

To do this in job executor, specify this at the end of the job executor command line command:

StageName:schema=*Protocol*:*SchemaFile*

In the Administration Utility, use the `--l` argument in the `job execute` command:

`--l` *StageName*:schema=*Protocol*:*SchemaFile*

Where:

StageName

The stage label shown under the stage's icon in the flow in Spectrum Enterprise Designer. For example, if the stage is labeled "Read from File" you would specify `Read from File` for the stage name.

To specify a stage within an embedded flow or a subflow, preface the stage name with the name of the embedded flow or subflow, followed by a period then the stage name:

EmbeddedOrSubflowName.*StageName*

For example, to specify a stage named Write to File in a subflow named Subflow1, you would specify:

```
Subflow1.Write to File
```

To specify a stage in an embedded flow that is within another embedded flow, add the parent flow, separating each with a period. For example, if Embedded Dataflow 2 is inside Embedded Dataflow 1, and you want to specify the Write to File stage in Embedded Dataflow 2, you would specify this:

```
Embedded Dataflow 1.Embedded Dataflow 2.Write to File
```

Protocol

A communication protocol:

file Use the file protocol if the file is on the same machine as the Spectrum Technology Platform server. For example, on Windows specify:

```
"file:/C:/myfile.txt"
```

On Linux specify:

```
"file:/testfiles/myfile.txt"
```

esclient Use the esclient protocol if the file is on the computer where you are executing the job if it is a different computer from the one running the Spectrum Technology Platform server. Use this format:

esclient:*ComputerName/path to file*

For example,

```
esclient:mycomputer/testfiles/myfile.txt
```

> **Note:** If you are executing the job on the server itself, you can use either the file or esclient protocol, but are likely to have better performance using the file protocol.

If the host name of the Spectrum Technology Platform server cannot be resolved, you may get the error "Error occurred accessing file". To resolve this issue, open this file on the server: *SpectrumDirectory*`/server/conf/spectrum-container.properties`. Set the `spectrum.runtime.hostname` property to the IP address of the server.

esfile Use the esfile protocol if the file is on a file server. The file server must be defined in Spectrum Management Console as a resource. Use this format:

esfile://*file server/path to file*

For example,

```
esfile://myserver/testfiles/myfile.txt
```

Where myserver is an FTP file server resource defined in Spectrum Management Console.

webhdfs Use the webhdfs protocol if the file is on a Hadoop Distributed File Server. The HDFS server must be defined in Spectrum Management Console as a resource. Use this format:

```
webhdfs://file server/path to file
```

For example,

```
webhdfs://myserver/testfiles/myfile.txt
```

Where myserver is an HDFS file server resource defined in Spectrum Management Console.

SchemaFile

The full path to the file that defines the layout you want to use.

> **Note:** You must use forward slashes in file paths. Do not use backslashes.

To create a schema file, define the layout you want in Read from File or Write to File, then click the **Export** button to create an XML file that defines the layout.

> **Note:** You cannot override a field's data type in a schema file when using job executor. The value in the <Type> element, which is a child of the FieldSchema element, must match the field's type specified in the flow's Read from File or Write to File stage.

Example File Format Override

This example executes a job named TestJob. Instead of writing the output to the file specified in the Write to File stage, it will write the output to `outputoverride.txt`. Instead of using the file schema specified in the Write to File stage in the flow, the job will use the schema specified in `output-data.xml`.

```
job execute --j TestJob --l "Write to
File=file:/Users/me/outputoverride.txt,Write to
File:schema=file:/Users/me/output-data.xml"
```

Spectrum Lineage & Impact Analysis

notes export

The `notes export` command exports Spectrum Lineage & Impact Analysis entity notes to a JSON file. Entity notes are the user-created notes in the Properties window of an entity on the Spectrum Lineage & Impact Analysis canvas.

Usage

`notes export --o` *OutputDirectory*

Required	Argument	Description
No	`--o` *OutputDirectory*	Specifies the directory to which you want to export the Spectrum Lineage & Impact Analysis notes. The path you specify here is relative to the directory where you are running the Administration Utility. If you omit this argument, the dataflow is exported to the directory containing the Administration Utility.

notes import

The `notes import` command imports Spectrum Lineage & Impact Analysis entity notes from a file created with the `notes export` command. Existing notes will be overwritten.

Usage

`notes import --f` *NotesFile*

Required	Argument	Description
Yes	`--f` *NotesFile*	Specifies the JSON file containing the Spectrum Lineage & Impact Analysis notes you want to import.

Spectrum Machine Learning

binning delete

This command deletes a binning from the server.

A binning cannot be deleted if it is exposed.

Usage

`binning delete` --binningname "*BinningName*"

Required	Argument	Description
Yes	`--n` *BinningName*	Specifies the name of the binning to delete from the server.

Example

This example deletes the binning named Home Sequence.

```
binning delete --n "Home Sequence"
```

binning expose

This command exposes the binning to make it available to the Binning Lookup stage.

After you expose the binning it becomes available for lookup and may not be deleted.

Usage

`binning expose --n` *binningname*

Required	Argument	Description
Yes	`--n` *binningname*	Specifies the name of the binning to expose on the server.

Example

This example exposes the binning named Home Sequence.

```
binning expose --n "Home Sequence"
```

binning list

The `binning list` command shows a list of all binnings on the server in alphabetical order.

Usage

```
binning list
```

binning unexpose

This command unexposes the binning to make it unavailable to the Binning Lookup stage.

After you unexpose the binning it becomes unavailable for lookup and may also be deleted.

Usage

`binning unexpose --n` *binningname*

Required	Argument	Description
Yes	`--n` *binningname*	Specifies the name of the binning to unexpose on the server.

Example

This example unexposes the binning named Home Sequence.

```
binning unexpose --n "Home Sequence"
```

machinelearning model expose

This command makes a machine learning model available to the Java Model Scoring stage.

A machine learning model must be exposed to be used for scoring.

Usage

`machinelearning model expose --m` *model*

Required	Argument	Description
Yes	`--m` *model*	Specifies the machine learning model.

Example

This example exposes model called ConsumerFraud.

```
machinelearning model expose --m ConsumerFraud
```

machinelearning model unexpose

This command makes a machine learning model unavailable to the Java Model Scoring stage.

An unexposed machine learning model cannot be used for scoring.

Usage

`machinelearning model unexpose --m` *model*

Required	Argument	Description
Yes	`--m` *model*	Specifies the machine learning model.

Example

This example unexposes model called ConsumerFraud.

```
machinelearning model unexpose --m ConsumerFraud
```

Match Keys

matchkey list

The `matchkey list` command lists all the match keys on the server. For each match key, the match key name is displayed. For more information about match keys, see *Match Key Generator* under the *Accessing Stages through Spectrum Flow Designer* section of the Spectrum Technology Platform *Data Quality Guide*.

Usage

```
matchkey list
```

matchkey delete

The `matchkey delete` command removes a match key from your system. For more information about match keys, see *Match Key Generator* under the *Accessing Stages through Spectrum Flow Designer* section of the Spectrum Technology Platform *Data Quality Guide*.

Usage

`matchkey delete` ***matchKeyName***

Required	Argument	Description
Yes	`-- d` *matchKeyName*	Specifies the match key to delete.

Example

This example deletes the match key named *My Match Key*.

```
matchkey delete My Match Key
```

matchkey export

The `matchkey export` command exports a match key that was created using Spectrum Flow Designer. The match key can then be imported to another server. You can export the match key as a JSON file. For more information about match keys, see *Match Key Generator* under the *Accessing Stages through Spectrum Flow Designer* section of the Spectrum Technology Platform *Data Quality Guide*.

Usage

`matchkey export` ***matchKeyName*** `--o` ***OutputDirectory***

Required	Argument	Description
Yes	`--m` *matchKeyName*	Specifies the name of the match key you want to export. **Note:** If you are unsure of the exact match key name, you can use the `matchkey list` command to get a list of the match key names.
No	`--o` *OutputDirectory*	Specifies the directory to which you want to export the match key. The path you specify here is relative to the directory where you are running the Administration Utility. If you omit this argument, the match key is exported to the directory containing the Administration Utility.

Example: Match key export

This example exports a match key named *My Match Key* in JSON format to a subfolder named `export` in the location where you are running the Administration Utility.

```
matchkey export My Match Key --o export
```

matchkey import

The `matchkey import` command imports a match key file into the server. Match keys are created using the *Match Key Generator* stage in Spectrum Flow Designer. For more information about match keys, see *Match Key Generator* under the *Accessing Stages through Spectrum Flow Designer* section of the Spectrum Technology Platform *Data Quality Guide*.

Usage

`matchkey import` *MatchKey* `--u` *TrueOrFalse*

Required	Argument	Description
Yes	`--f` *MatchKeyFile*	Specifies the match key file you want to import. Directory paths you specify here are relative to the location where you are running the Administration Utility.

Required	Argument	Description
No	`--u` *TrueOrFalse*	Specifies whether to overwrite the existing match key file if a match key file with the same name is already on the server, where *TrueOrFalse* is one of these: **true** If there is a match key file on the server with the same name as the match key file you are importing, the match key file on the server will be overwritten. This is the default setting. **false** If there is a match key file on the server with the same name as the match key file you are importing, the match key file will not be imported.

Example

This example imports the match key named `MyMatchKey.json`, which is located in a subfolder named `exported` in the location where you are running the Administration Utility. Because no `--u` command is specified, any existing match key file of the same name on the server will be overwritten.

```
matchkey import exported\MyMatchKey.json
```

Match Rules

matchrule delete

The `matchrule delete` command removes a match rule from your system. For more information, see the "Matching" section of the *Data Quality Guide*.

Usage

`matchrule delete` *matchRuleName*

Required	Argument	Description
Yes	`--d` *matchRuleName*	Specifies the match rule to delete.

Example

This example deletes the match rule named My Match Rule.

```
matchrule delete My Match Rule
```

matchrule export

The `matchrule export` command exports a match rule that was created using one of the matching stages (Interflow Match, Intraflow Match, Transactional Match) in the Spectrum Enterprise Designer. The match rule can then be imported to another server. You can export the match rule as *.mr* or *.json* files.

For more information, see the "Matching" section of the *Data Quality Guide*.

Usage

`matchrule export` *matchRuleName* `--o` *OutputDirectory*

Required	Argument	Description
Yes	`--m` *matchRuleName*	Specifies the name of the match rule you want to export. **Tip:** If you are unsure of the exact match rule name you can use the `matchrule list` command to get a list of the match rule names.
No	`--j` *matchRuleName*	Specifies the name of the match rule you want to export in JSON file format. **Tip:** If you are unsure of the exact match rule name you can use the `matchrule list` command to get a list of the match rule names.
No	`--o` *OutputDirectory*	Specifies the directory to which you want to export the match rule. The path you specify here is relative to the directory where you are running the Administration Utility. If you omit this argument, the match rule is exported to the directory containing the Administration Utility.

Example: Match rule export

This example exports a match rule named "My Match Rule" in *mr* format to a subfolder named `export` in the location where you are running the Administration Utility.

```
matchrule export My Match Rule --o export
```

Example: Match rule export in JSON format

This example exports a match rule named "My Match Rule" in *JSON* format to the directory containing the Administration Utility.

```
matchrule export My Match Rule --j
```

matchrule import

The `matchrule import` command imports a match rule file into the server. Match rules are created using one of the matching stages (Interflow Match, Intraflow Match, Transactional Match) in Spectrum Enterprise Designer. For more information, see the "Matching" section of the *Data Quality Guide*.

Usage

`matchrule import` *MatchRule* `--u` *TrueOrFalse*

Required	Argument	Description
Yes	`--f` *MatchRuleFile*	Specifies the match rule file you want to import. Directory paths you specify here are relative to the location where you are running the Administration Utility.
No	`--u` *TrueOrFalse*	Specifies whether to overwrite the existing match rule file if a match rule file with the same name is already on the server, where *TrueOrFalse* is one of these: **true** If there is a match rule file on the server with the same name as the match rule file you are importing, the match rule file on the server will be overwritten. This is the default setting. **false** If there is a match rule file on the server with the same name as the match rule file you are importing, the match rule file will not be imported.

Example

This example imports the match rule named `MyMatchRule.mr`, which is located in a subfolder named `exported` in the location where you are running the Administration Utility. Because no --u command is specified, any existing match rule file of the same name on the server will be overwritten.

```
matchrule import exported\MyMatchRule.mr
```

matchrule list

The `matchrule list` command lists all the match rules on the server. For each match rule, the following information is displayed: the match rule name and whether the match rule is exposed. For more information, see the "Matching" section of the *Data Quality Guide*.

Usage

```
matchrule list
```

Best of Breed Rules

bestofbreedrule delete

The `bestofbreedrule delete` command removes a best of breed rule from your system. For more information, see the "Matching" section of the *Data Quality Guide*.

Usage

`bestofbreedrule delete` *bobRuleName*

Required	Argument	Description
Yes	`--d` *bobRuleName*	Specifies the best of breed rule to delete.

Example

This example deletes the best of breed rule named My BOB Rule.

```
bestofbreedrule delete My BOB Rule
```

bestofbreedrule export

The `bestofbreedrule export` command exports a best of breed rule that was created using the **Best of Breed** stage in the Spectrum Enterprise Designer. The best of breed rule can then be imported to another server. You can export the best of breed rule as a *.bob*file.

For more information, see the "Matching" section of the *Data Quality Guide*.

Usage

`bestofbreedrule export` *bobRuleName* `--o` *OutputDirectory*

Required	Argument	Description
Yes	`--b` *bobRuleName*	Specifies the name of the best of breed rule you want to export. **Tip:** If you are unsure of the exact best of breed rule name, you can use the `bestofbreedrule list` command to get a list of the best of breed rule names.
No	`--o` *OutputDirectory*	Specifies the directory to which you want to export the best of breed rule. The path you specify here is relative to the directory where you are running the Administration Utility. If you omit this argument, the match rule is exported to the directory containing the Administration Utility.

Example: Best of breed rule export

This example exports a best of breed rule named "My BOB Rule" in *.bob* format to a subfolder named `export` in the location where you are running the Administration Utility.

```
bestofbreedrule export My BOB Rule --o export
```

bestofbreedrule import

The `bestofbreedrule import` command imports a best of breed rule file into the server. Best of breed rules are created using the **Best of Breed** stage in Spectrum Enterprise Designer. For more information, see the "Matching" section of the *Data Quality Guide*.

Usage

`bestofbreedrule import` *bobRuleName* `--u` *TrueOrFalse*

Required	Argument	Description
Yes	`--f` *bobRuleFile*	Specifies the best of breed rule file you want to import. Directory paths you specify here are relative to the location where you are running the Administration Utility.
No	`--u` *TrueOrFalse*	Specifies whether to overwrite the existing best of breed rule file if a best of breed rule file with the same name is already on the server, where *TrueOrFalse* is one of these: **true**

Required	Argument	Description
		If there is a best of breed rule file on the server with the same name as the best of breed rule file you are importing, the best of breed rule file on the server will be overwritten. This is the default setting.
	false	If there is a best of breed rule file on the server with the same name as the best of breed rule file you are importing, the best of breed rule file will not be imported.

Example

This example imports the best of breed rule named `MyBOBRule.bob`, which is located in a subfolder named `exported` in the location where you are running the Administration Utility. Because no --u command is specified, any existing best of breed rule file of the same name on the server will be overwritten.

```
bestofbreedrule import exported\MyBOBRule.bob
```

bestofbreedrule list

The `bestofbreedrule list` command lists all the best of breed rules on the server. For each best of breed rule, the following information is displayed: the best of breed rule name and whether the best of breed rule is exposed. For more information, see the "Matching" section of the *Data Quality Guide*.

Usage

```
bestofbreedrule list
```

Metadata Connections

resourceconnection export

The `resourceconnection export` command exports a resource connection from Spectrum server to a JSON file.

Usage

`resourceconnection export --n` *ConnectionName* `--o` *OutputDirectory*

Required	Argument	Description
Yes	`--n` *ConnectionName*	Specifies the name of the resource connection you want to export. If the connection name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the connection name, you can use the `resourceconnection list` command to get a list of the connection names.
Yes	`--t` *ConnectionType*	Specifies the connection type. It can have values, such as MSSQLServer, Oracle, Salesforce, and Apache Cassandra.
No	`--o` *OutputDirectory*	Specifies the directory to which you want to export the connection. The path you specify here is relative to the directory where you are running the Administration Utility. If you omit this argument, the connection is exported to the directory containing the Administration Utility.
No	`--encrypt` *fileEncryption*	Specifies if the file encryption is on or off. **true** Enables file encryption. This is the default setting. **false** Disables file encryption.

Example

This example exports the resource connection named "FrameallSalesforce_bulkon" of "SalesForce" connection type to a folder named `exported` which is a subfolder in the location where you have installed the Administration Utility.

```
resourceconnection export --n "FrameallSalesforce_bulkon" --t
SalesForce --o exported
```

resourceconnection import

The `resourceconnection import` command imports a database connection to the Spectrum server.

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

Usage

`resourceconnection import --f` *connectionJsonFileName*

Note: To see a list of parameters, type `help resourceconnection import`.

Required	Argument	Description
Yes	`--f` *file*	Specifies the connection JSON file name you want to import.
No	`--u` *update*	Specifies whether to overwrite the existing connection, if connection with the same name already exists on the server. **true** If there is a resource on the server with the same name as the resource you are importing, the resource on the server will be overwritten. **false** If there is a resource on the server with the same name as the resource you are importing, the resource will not be imported. This is the default setting.

Example

This example imports the connection "DynamicsTrial.json" and updates the existing connection if it is already present on the server.

```
resourceconnection import --f D:\trunk_19\DynamicsTrial.json
--u true
```

resourceconnection list

The `resourceconnection list` command returns a list of all the resource connections defined on Spectrum Technology Platform server.

Usage

```
resourceconnection list
```

Required	Argument	Description
No	`--t` *connectionType*	Specifies the type of the connection.

Example

This example displays list of all the connections configured in Spectrum Technology Platform.

```
resourceconnection list
```

Notification

notification daystoexpire set

The `notification daystoexpire set` command specifies the number of days in advance that you want to be notified of a pending license or data expiration.

Usage

```
notification daystoexpire set --d Days
```

Required	Argument	Description
Yes	`--d` *Days*	Specifies the number of days in advance that you want to be notified of a pending license or data expiration.

Example

This example sets notifications to be sent 30 days before a license or data expires.

```
notification daystoexpire set --d 30
```

notification enabled set

The `notification enabled set` command enables or disables email notifications of pending license or data expiration.

Usage

```
notification enabled set --e TrueOrFalse
```

Required	Argument	Description
Yes	`--e` *TrueOrFalse*	Enables or disables license expiration notification where *TrueOrFalse* is one of the following: **true** Enables license expiration notification. **false** Disables license expiration notification.

Example

This example enables notifications:

```
notification enabled set --e true
```

notification expirationsettings list

The `notification expirationsettings list` command displays the expiration notification settings in effect on your system. The command displays the number of days before expiration that notifications will be sent, the users who are subscribed to notifications, and whether notifications are enabled or disabled.

Usage

```
notification expirationsettings list
```

notification smtp get

The `notification smtp get` command displays the email settings used to send license expiration notifications.

Usage

```
notification smtp get
```

notification smtp set

The `notification smtp set` command specifies the email settings to use to send license expiration notification emails.

Usage

`notification smtp set --h` *Host* `--o` *Port* `--u` *UserName* `--p` *Password* `--e` *FromEmail*

Required	Argument	Description
No	`--h` *Host*	Specifies the host name or IP address of the SMTP server to use to send the notification email.
No	`--o` *Port*	Specifies the port number or range used by the SMTP server. The default is 25.
No	`--u` *UserName*	Specifies the user name to use to send the email. This must be a valid user account on the SMTP server.
No	`--p` *Password*	Specifies the password for the user account specified in the `--u` parameter.
No	`--e` *FromEmail*	Specifies the email address from which the notification email will be sent.

Example

This example sets SMTP settings for email notifications.

```
notification smtp set --h mail.example.com --o 25 --u me123
--p MyPassword --e spectrum@example.com
```

notification smtp test

The `notification smtp test` command sends a test email to an email address you specify. Use this command to verify the SMTP settings used for license expiration notification.

Usage

`notification smtp test --e` *DestinationEmail*

Required	Argument	Description
Yes	`--e` *DestinationEmail*	Specifies the email address to which you want to send a test email message.

Example

This example sends a test email address to `me@example.com`.

```
notification smtp test --e me@example.com
```

notification subscriber add

The `notification subscriber add` command adds an email address to receive license expiration notifications.

Usage

`notification subscriber add --e` *Email*

Required	Argument	Description
Yes	`--e` *Email*	Specifies an email address to receive license expiration notifications.

Example

This example adds the email address `me@example.com` to receive license expiration notifications.

```
notification subscriber add --e me@example.com
```

notification subscriber delete

The `notification subscriber delete` command removes an email address from the list of email addresses that receive license expiration notifications.

Usage

`notification subscriber delete --e` *Email*

Required	Argument	Description
Yes	`--e` *Email*	Specifies an email address to remove from the list of email addresses that receive license expiration notifications.

Example

This example removes the email address `me@example.com` from the list of email addresses that receive license expiration notifications.

```
notification subscriber delete --e me@example.com
```

Open Parser Cultures

openparser culture export

The `openparser culture export` command exports all Open Parser cultures. The cultures can then be imported to another server. For more information, see the "Parsing" section of the *Data Quality Guide*.

Usage

`openparser culture export` *OutputFile*

Required	Argument	Description
No	`--f` *OutputFile*	Specifies the file containing the cultures you want to export. If you omit this argument, the Administration Utility automatically exports the cultures.xml file.

Example

This example exports the `culturesFR.xml` file.

```
openparser culture export culturesFR.xml
```

openparser culture import

The `openparser culture import` command imports an Open Parser culture file into the server. For more information, see the "Parsing" section of the *Data Quality Guide*.

Usage

`openparser culture import` *CultureFileName*

Required	Argument	Description
Yes	`--f` *CultureFileName*	Specifies the file containing the culture you want to import.

Example

This example imports the file named `MyCulture.xml`.

```
openparser culture import MyCulture.xml
```

Open Parser Domains

openparser domain export

The `openparser domain export` command exports an Open Parser domain. The domain can then be imported to another server. For more information, see the "Parsing" section of the *Data Quality Guide*.

Usage

`openparser domain export` *DomainName* `--o` *OutputDirectory*

Required	Argument	Description
Yes	`--d` *DomainName*	Specifies the name of the domain you want to export. **Tip:** If you are unsure of the exact domain name you can use the `openparser domain list` command to get a list of the domain names.
No	`--o` *OutputDirectory*	Specifies the directory to which you want to export the domain. The path you specify here is relative to the directory where you are running the Administration Utility. If you omit this argument, the domain is exported to the directory containing the Administration Utility.

Example

This example exports a domain named "MyDomain" to a folder named `exported`, which is a subfolder in the location where you have installed the Administration Utility.

```
openparser domain export MyDomain --o exported
```

openparser domain import

The `openparser domain import` command imports an Open Parser domain into the server. For more information, see the "Parsing" section of the *Data Quality Guide*.

Usage

`openparser domain import` *DomainFileName*

Required	Argument	Description
Yes	`--f` *DomainFileName*	Specifies the file containing the domain you want to import. Directory paths you specify here are relative to the location where you are running the Administration Utility.

Example

This example imports the file named `MyDomain.xml`, which is located in a subfolder named `exported` in the location where you are running the Administration Utility.

```
openparser domain import exported\MyDomain.xml
```

openparser domain list

The `openparser domain list` command lists all the Open Parser domains on the server. For each domain, the following information is displayed: the domain name and whether the domain is exposed.

Usage

```
openparser domain list
```

Performance Monitor

performancemonitor enabled get

The `performancemonitor enabled get` command displays the jobs and services that have performance monitoring enabled.

Usage

```
performancemonitor enabled get
```

performancemonitor enabled set

The `performancemonitor enabled set` command enables and disables performance monitoring for a job or service. When performance monitoring is enabled, performance information for the job or service is written to the performance log. The performance log includes overall performance information for the job or service as well as performance information for the job or service dataflow.

Usage

`performancemonitor enabled set --e` *TrueOrFalse* `--d` *DataflowName*

Required	Argument	Description
Yes	`--e` *TrueOrFalse*	Enables or disables performance monitoring where *TrueOrFalse* is one of the following: **true** Enables performance monitoring. **false** Disables disables performance monitoring.
Yes	`--d` *DataflowName*	Specifies the name of the dataflow for which you want to enable or disable performance monitoring. **Tip:** If you are unsure of the exact dataflow name you can use the `dataflow list` command to get a list of the dataflow names.

Example

This example turns on performance monitoring for the dataflow MyNameParsingDataflow:

```
performancemonitor enabled set --e true --d
"MyNameParsingDataflow"
```

The Performance Log

The performance log contains details about how long it takes for a job or service to run. It includes overall performance information for the job or service as well as performance information for each stage in the job or service dataflow. You can use this information to identify bottlenecks in your dataflow by looking at the execution time and processing time for each stage. A large difference between execution time and processing time means that the stage is spending time waiting for data from upstream stages. This may indicate that an upstream stage is a bottleneck in the dataflow. Note that for sinks, a large difference between execution time and processing time does not necessarily indicate a performance problem because sinks typically have to wait for the first records from the rest of the dataflow.

To enable performance monitoring for a job or service, use the `performancemonitor enabled set` command in the Administration Utility.

The performance log is located on your Spectrum Technology Platform server in the following location:

SpectrumDirectory`\server\logs\performance.log`

The performance log contains one row for each run of a monitored job or service. It is a rolling log that consists of a maximum of five files. Each file is limited to 10 MB in size. Once this limit is reached, the oldest performance data is deleted when new performance data is logged.

> **Note:** The log file path name, maximum file size, and maximum number of files are specified by the PERFORMANCE `appender` settings in the `logback.xml` configuration file. For more information, see **Logback configuration file** on page 246.

Each entry in the performance log contains the following information. For ease of reading, line breaks and indentation are shown below. In the actual log, the entry is on one line.

```
Date Time [performance]
{
     "username":"UserName",
     "dataflowId":"DataflowName",
     "runMode":"BatchOrRealTime",
     "elapsedTime":Nanoseconds,
     "stageInfo":[
          {
               "stageName":"Name",
               "stageLabel":"Label",
               "options":{
                    OptionsList
               },
               "recordsRead":Count,
               "recordsWritten":Count,
               "executionTime":Nanoseconds,
               "processingTime":Nanoseconds
               "readBlockingTime":Nanoseconds
               "writeBlockingTime":Nanoseconds
               "readBlockingPercent":Percentage
               "writeBlockingPercent":Percentage
          }
     ]
}
```

Where:

username

The user who executed the job or service.

dataflowID

The name of the service or job as defined in Spectrum Enterprise Designer.

runMode

Indicates whether the log entry is for a job or a service. One of the following:

Batch	The log entry is for a job.
RealTime	The log entry is for a service.

elapsedTime

The time in nanoseconds that it took to run the job or service request.

stageInfo

Lists runtime information for each stage in the dataflow. The following information is listed for each stage:

stageName

The permanent name of the stage.

stageLabel

The user-defined name of the stage. The stage label is shown on the canvas in Spectrum Enterprise Designer.

options

If any options were specified at runtime, those options and their settings are listed here.

recordsRead

The total number of records that passed into the stage through all the stage's input ports.

recordsWritten

The total number of records that the stage wrote to all its output ports.

executiontime

The amount of time from when the stage processed its first record to when it processed its last record. This includes the time the stage was idle while waiting for data from other stages in the dataflow.

processingtime

The amount of time the stage spent actively processing records, not including the time it was idle while waiting for other stages in the dataflow.

readBlockingTime

The amount of time blocking while waiting to read the next record. A high read blocking time means that a prior process is taking longer than this stage and may indicate additional tuning is needed.

writeBlockingTime

The amount of time blocking while waiting to write the next record. A high write blocking time means that a prior process is taking longer than this stage and may indicate additional tuning is needed.

readBlockingPercent

The percentage of the total execution time that a stage was blocking on reading a record.

writeBlockingPercent

the percentage of the total execution time that a stage was blocking on writing a record.

Permissions

permission list

The `permission list` command lists the names of the entities to which you can assign permissions.

Usage

```
permission list
```

Physical and Logical Models

logicalmodel bulkExport

The `logicalmodel bulkExport` command exports all the logical models and their metadata from Discovery to the specified directory. If you do not specify the output directory, the models are exported to the directory from which you are running the command. To export the logical models along with the dependent physical models, use the *exportDependency* argument.

Usage

`logicalmodel bulkExport --o` *outputDirectory* `--d` *exportDependency trueOrFalse*

Required	Argument	Description
No	`--o` *outputDirectory*	Specifies the directory where the logical models are to be exported.
No	`--d` *exportDependency*	Specifies if the models are to be exported along with all their dependencies. **true** Models exported along with dependencies. **false** Models exported without the dependent models

Example

This example exports all the logical models along with the dependencies to the "MyModelExport" folder, located at: `C:\Spectrum\LogicalModels`.

```
logicalmodel bulkExport --o
C:\Spectrum\LogicalModels\MyModelExport --d true
```

logicalmodel bulkImport

The `logicalmodel bulkImport` command imports all the logical models and their metadata from the specified directory to Discovery. To import the logical models along with their dependent physical models, use the *importDependency* argument.

Usage

`logicalmodel bulkImport --i` *inputDirectory* `--u` *trueOrFalse* `--d` *trueOrFalse*

Required	Argument	Description
No	`--i` *inputDirectory*	Specifies the directory from which the logical models are to be imported.
No	`--u` *updateIfExists*	Specifies whether to update the existing model. **true** updates the existing logical models. **false** Does not update the existing logical models.
No	`--d` *importDependency*	Specifies whether to import the logical models along with their dependencies. **true** Logical models imported along with dependencies. **false** Logical models imported without the dependent models

Example

This example imports all the logical models along with their dependent models to the "MyModel" folder located here: `C:\Spectrum\LogicalModels`. It also updates the existing model of the same name.

```
logicalmodel bulkImport --i C:\Spectrum\LogicalModels\MyModel
--u true --d true
```

logicalmodel export

The `logicalmodel export` command exports the specified logical model and its metadata from Discovery to the specified directory. If you do not specify the output directory, the model is exported to the directory from which you are running the command. To export the logical model along with the dependent physical models, use the *exportDependency* argument.

Usage

`logicalmodel export --n` *logicalModelName* `--o` *outputDirectory* `--d` *trueOrFalse*

Required	Argument	Description
Yes	`--n` *logicalModelName*	Specifies the name of the logical model you want to export. Directory paths you specify here are relative to the location where you are running the Administration Utility.
No	`--o` *outputDirectory*	Specifies the location of the folder that will store the exported logical model.
No	`--d` *exportDependency*	Specifies whether to export the logical model along with its dependencies. **true** Model exported along with dependencies. **false** Model exported without the dependent models

Example

This example exports the logical model "MyModel", along with all the dependencies, to the "MyModelExport" folder located at: `C:\Spectrum\LogicalModels`.

```
logicalmodel export --n MyModel --o
C:\Spectrum\logicalModels\MyModelExport --d true
```

logicalmodel import

The `logicalmodel import` command imports the specified logical model and its metadata to Discovery. To import the logical model along with its dependent physical models, use the *importDependency* argument.

Usage

`logicalmodel import --i` *logicalModelInputFile* `--u` *trueOrFalse* `--d` *trueOrFalse*

Required	Argument	Description
Yes	`--i` *logicalModelInputFile*	Specifies the logical model file to be imported.
No	`--u` *updateIfExists*	Specifies whether to update the existing model with same name in Discovery with the imported model. **true** updates the existing logical model. **false** Does not update the existing logical model.

Required	Argument	Description	
No	`--d` *importDependency*	Specifies whether to import the logical model along with its dependencies.	
		true	Logical model imported along with dependencies.
		false	Logical model imported without the dependent models

Example

This example imports the logical model file "MyModel", along with its dependent models to Discovery and updates the already existing file with this one.

```
logicalmodel import --i MyModel --u true --d true
```

logicalmodel list

The `logicalmodel list` command returns a list of all logical models.

Usage

```
logicalmodel list
```

Example

This example lists all logical models.

```
logicalmodel list
```

modelstore bulkExport

The `modelstore bulkExport` command exports all the model stores from Discovery.

Usage

`modelstore bulkExport --o` *outputDirectory* `--d` *trueOrFalse*

Required	Argument	Description
No	`--o` *outputDirectory*	Specifies the directory to which the model stores are to be exported.

Required	Argument	Description	
No	`--d` *exportDependency*	Specifies if the model stores are to be exported along with their dependencies.	
		true	Model stores exported along with dependencies.
		false	Model stores exported without the dependent models

Example

This example exports all the model stores, along with their dependencies to the "MyModelStoreExport" folder at: `C:\Spectrum\ModelStore`.

```
modelstore bulkExport --o
C:\Spectrum\ModelStore\MyModelStoreExport --d true
```

modelstore deploy

The `modelstore deploy` command deploys the specified model store to the Spectrum server.

Usage

`modelstore deploy --n` *modelstoreName*

Required	Argument	Description
Yes	`--n` *modelStoreName*	Specifies the name of the model store to be deployed.

Example

This example deploys the model store named "MyModelStore" to Spectrum server.

```
modelstore deploy --n MyModelStore
```

modelstore export

The `modelstore export` command exports the specified model store from Discovery to the specified folder.

Usage

`modelstore export --n` *modelStoreName* `--o` *outputDirectory* `--d` *trueOrFalse*

Required	Argument	Description
Yes	`--n` *modelStoreName*	Specifies the name of the logical model to export.
No	`--o` *outputDirectory*	Specifies the directory in which the exported model stores are to be saved. If you do not specify this path, the model store is saved to the directory from which you are running the command.
No	`--d` *exportDependency*	Specifies if the model store is to be exported along with its dependencies. **true** Model store exported along with dependencies. **false** Model store exported without the dependent models

Example

This example exports the model store "MyModelStore", along with all its dependencies to the "MyModelStore" folder located here: `C:\Spectrum\ModelStores`

```
modelstore export --n MyModelStore --o C:\Spectrum\ModelStores
--d true
```

modelstore import

The `modelstore import` command imports the specified model store file to the Discovery.

Usage

`modelstore import --i` *modelstoreInputFile* `--u` *trueOrFalse*`--d` *trueOrFalse*

Required	Argument	Description
Yes	`--i` *modelstoreInputFile*	Specifies the model store file to be imported.
No	`--u` *updateIfExists*	Specifies the existing model store (with same name) is to be updated with the imported one. **true** updates the existing model store. **false** Does not update the existing model store.

Required	Argument	Description
No	`--d` *importDependency*	Specifies the model store is to be imported along with its dependencies. **true** Model store imported along with dependencies. **false** Model store imported without the dependent models

Example

This example imports the model store file "MyModelStore", along with its dependencies and updates the already existing model store with this name.

```
modelstore import --i MyModelStore --u --d
```

modelstore bulkImport

The `modelstore bulkImport` command imports all the model stores to Discovery.

Usage

`modelstore bulkImport --i` *inputDirectory* `--u` *trueOrFalse* `--d` *trueOrFalse*

Required	Argument	Description
No	`--i` *inputDirectory*	Specifies the location of the folder from which the model stores are to be imported.
No	`--u` *updateIfExists*	Specifies whether to update the existing model stores with same names by the imported model stores. **true** updates the existing model stores. **false** Does not update the existing model stores.
No	`--d` *importDependency*	Specifies that the model stores are to be imported along with all the dependencies. **true** Model stores imported along with dependencies. **false** Model stores imported without the dependent models

Example

This example imports all the model stores, along with their dependencies from the "MyModel" folder located here: `C:\Spectrum\modelstore` to Discovery. It also overwrites any existing model by the same name.

```
modelstore bulkImport --i C:\Spectrum\modelstore\MyModel --u
true --d true
```

modelstore list

The `modelstore list` command returns a list of all the model stores.

Usage

```
modelstore list
```

Example

This example lists all model stores.

```
modelstore list
```

modelstore undeploy

The `modelstore undeploy` command undeploys the specified model store from the Spectrum server.

Usage

`modelstore undeploy --n` *modelstoreName*

Required	Argument	Description
Yes	`--n` *modelstoreName*	Specifies the name of the model store to be undeployed.

Example

This example undeploys the model store "MyModelStore".

```
modelstore undeploy --n MyModelStore
```

physicalmodel bulkExport

The `physicalmodel bulkExport` command exports all the physical models and their metadata from Discovery to the specified directory.

Usage

`physicalmodel bulkExport --o` *outputDirectory*

Required	Argument	Description
No	`--o` *outputDirectory*	Specifies the directory where the physical models are to be exported. If unspecified, the models are exported to the directory from which you run the command.

Example

This example exports all the physical models to the "MyModelExport" folder located at: `C:\Spectrum\PhysicalModels`.

```
physicalmodel bulkExport --o
C:\Spectrum\PhysicalModels\MyModelExport
```

physicalmodel bulkImport

The `physicalmodel bulkImport` command imports all the physical models to Discovery from the specified input directory.

Usage

`physicalmodel bulkImport --i` *inputDirectory*

Required	Argument	Description
No	`--i` *inputDirectory*	Specifies the directory from which the physical models are to be imported.
No	`--u` *updateIfExists*	Specifies that the existing models with same names are to be updated by the imported models.

Example

This example imports all the physical models to the "MyModel" folder located here: `C:\Spectrum\PhysicalModels`. It also overwrites any existing model of the same name.

```
physicalmodel bulkImport --i
C:\Spectrum\PhysicalModels\MyModel --u
```

physicalmodel export

The `physicalmodel export` command exports the specified physical model and its metadata from Discovery to the specified directory.

Usage

`physicalmodel export --n` *physicalModelName* `--o` *outputDirectory*

Required	Argument	Description
Yes	`--n` *physicalModelName*	Specifies the name of the physical model to be exported.
No	`--o` *outputDirectory*	Specifies the directory to which the models are exported. If unspecified, the models are exported to the directory from which you are running the command.

Example

This example exports the physical model "MyModel" to the "MyModelExport" folder located here: `C:\Spectrum\PhysicalModels`.

```
physicalmodel export --n MyModel --o
C:\Spectrum\PhysicalModels\MyModelExport
```

physicalmodel import

The `physicalmodel import` command imports the specified physical model file and its metadata to Discovery.

Usage

`physicalmodel import --i` *physicalModelInputFile* `--u` *trueOrFalse*

Required	Argument	Description
Yes	`--i` *physicalModelInputFile*	Specifies the physical model file to be imported.
No	`--u` *updateIfExists*	Specifies that the imported model is to update the existing model of same name. **true** updates the existing physical model. **false** Does not update the existing physical model.

Example

This example imports the physical model file "MyModel" and updates the already existing file with this one.

```
physicalmodel import --i MyModel --u true
```

physicalmodel list

The `physicalmodel list` command returns a list of all the physical models.

Usage

`physicalmodel list --t` *dataSourceType*

Required	Argument	Description
No	`--t` *dataSourceType*	Specifies the datasource type of the physical models to be listed. **Note:** If unspecified, physical models of all types in Discovery are listed.

Example

This example lists all physical models of Salesforce.

```
physicalmodel list --t Salesforce
```

This example lists all physical models in Discovery.

```
physicalmodel list
```

Process Flows

processflow delete

The `processflow delete` command removes a process flow from your system.

Usage

`processflow delete --n` *ProcessFlowName*

Required	Argument	Description
Yes	`--n` *ProcessFlowName*	Specifies the process flow to delete. If the process flow name contains spaces, enclose the process flow name in quotes.

Example

This example deletes the process flow named My Process Flow.

```
processflow delete --n "My Process Flow"
```

processflow execute

The `processflow execute` command runs one or more process flows. This command is one of two ways you can execute process flows from the command line. The other way is to use the Process Flow Executor, which is a command line utility available from the Spectrum Technology Platform welcome page on your server. The advantage of using the `processflow execute` command in the Administration Utility is that it allows you to also include other commands in a single script or batch file. For example, you could expose the process flow using the `processflow expose` command then execute it using the `processflow execute` command. The `processflow execute` command provides the same features as the Process Flow Executor.

Usage

`processflow execute --r` *ProcessFlowNames* `--f` *propertyFile* `--i` *PollInterval* `--d` *DelimiterCharacter* `--t` *Timeout* `--w` *WaitToComplete* `--o` *StageName=File*

Required	Argument	Description
No	`--?`	Prints usage information.
No	`--d` *DelimiterCharacter*	Sets a delimiter to use to separate the status information displayed in the command line when you run the command. The default is "|". For example, using the default character, the message below is displayed at the command line when you run a process flow named "MyProcessflow": `MyProcessflow|1|Succeeded`
No	`--f` *PropertyFile*	Specifies a path to a property file. For more information on property files, see **Using a Process Flow Property File** on page 208.
No	`--i` *PollInterval*	Specifies how often to check for completed jobs, in seconds. The default is "5".
Yes	`--r` *ProcessFlowNames*	A comma-separated list of process flows to run. Required. **Note:** If you specify any file overrides this argument must not be the last argument specified.
No	`--t` *Timeout*	This option is deprecated and will be ignored.
No	`--v` *Verbose*	Return verbose output where *Verbose* is one of the following: **true** Return verbose output. **false** Do not return verbose output. **Note:** If you specify any file overrides this argument must not be the last argument specified.
No	`--w` *WaitToComplete*	This option is deprecated and will be ignored.
No	`--o` *StageName=File*	Overrides the input or output file specified in the job. For more information, see **Overriding Process Flow File Locations** on page 387.

Example

This example executes the process flow named "My Process Flow".

```
processflow execute --r "My Process Flow"
```

Overriding Process Flow File Locations

When you run a process flow using the `process flow execute` command in the Administration Utility, you can specify that the process flow should use different input and output files than those specified in the job. To do this use the `--o` argument:

```
--o "JobName|StageName=File"
```

Where:

JobName

The name of a job referenced in the process flow.

StageName

The name of a Read from File or Write to File stage in the job as shown in the stage label under the stage's icon in the dataflow. For example, if the input stage is labeled "Read From File" you would specify:

```
"Job1|Read From File=file:C:/inputfile.txt"
```

If the input stage is labeled "Illinois Customers" you would specify:

```
"Job1|Illinois Customers=file:C:/inputfile.txt"
```

File

The protocol and full path to the file. You must use forward slashes in file paths, not backslashes. The protocol must be one of these:

file:

If the file is on the same machine as the Spectrum Technology Platform server, start the path with the "file:" protocol. For example, on Windows specify `file:C:/myfile.txt` and on Linux specify `file:/testfiles/myfile.txt`.

> **Note:** If the client and server are running on the same machine, you can use either the "file:" or "esclient:" protocol, but are likely to have get better performance using the "file:" protocol.

esclient:

If the file is on the same machine as Process Flow Executor, start the path with the "esclient:" protocol. For example, on Windows specify `esclient:C:/myfile.txt` and on Linux specify `esclient:/testfiles/myfile.txt`.

> **Note:** If the machine running process flow executor cannot resolve the host name of the Spectrum Technology Platform server, you may get an error "Error occurred accessing file". To resolve this issue, open this file on the server: *SpectrumDirectory*`/server/conf/spectrum-container.properties`. Set the `spectrum.runtime.hostname` property to the IP address of the server.

ftp:

To use a file server defined in Spectrum Management Console, use this format: `ftp:`*`NameOfFileServer/PathToFile`*. For example, `ftp://FS/testfiles/myfile.txt` where FS is a file server resource defined in Spectrum Management Console.

This example illustrates how to override file locations using the `--o` argument:

```
--o "Job1|Read from File=file:C:/myfile_input.txt,Job1|Write to
File=file:C:/myfile_output.txt"
```

processflow export

The `processflow export` command exports a process flow from the server to a `.pf` file. The process flow can then be imported to another server.

Note: Process flows can only be exchanged between identical versions of Spectrum Technology Platform.

Usage

`processflow export --n` *ProcessFlowName* `--o` *OutputFile*

Required	Argument	Description
Yes	`--n` *ProcessFlowName*	Specifies the name of the process flow you want to export. If the process flow name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the exact process flow name you can use the `processflow list` command to get a list of the process flow names.
No	`--o` *OutputFile*	Specifies the directory to which you want to export the process flow. The path you specify here is relative to the directory where you are running the Administration Utility. If you omit this argument, the process flow is exported to the directory containing the Administration Utility.

Example

This example exports the process flow named "My Process Flow" to a folder named `exported` which is a subfolder in the location where you have installed the Administration Utility.

```
processflow export --n "My Process Flow" --o exported
```

processflow expose

The `processflow expose` command makes the process flow available for execution.

Note: If you use dataflow versioning in Spectrum Enterprise Designer, the `processflow expose` command exposes the most recent version of the dataflow.

Usage

`processflow expose --n` *ProcessFlowName*

Required	Argument	Description
Yes	`--n` *ProcessFlowName*	Specifies the name of the process flow you want to expose. If the process flow name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the exact process flow name you can use the `processflow list` command to get a list of the process flow names.

Example

This example exposes the process flow named "My Process Flow".

```
processflow expose --n "My Process Flow"
```

processflow import

The `processflow import` command imports a process flow file (a `.pf` file) into the server. Process flow files are created by exporting a process flow from the server using the `processflow export` command

Usage

`processflow import --f` *ProcessFlowFile* `--u` *TrueOrFalse* `--p` *Path* `--c` *TrueOrFalse*

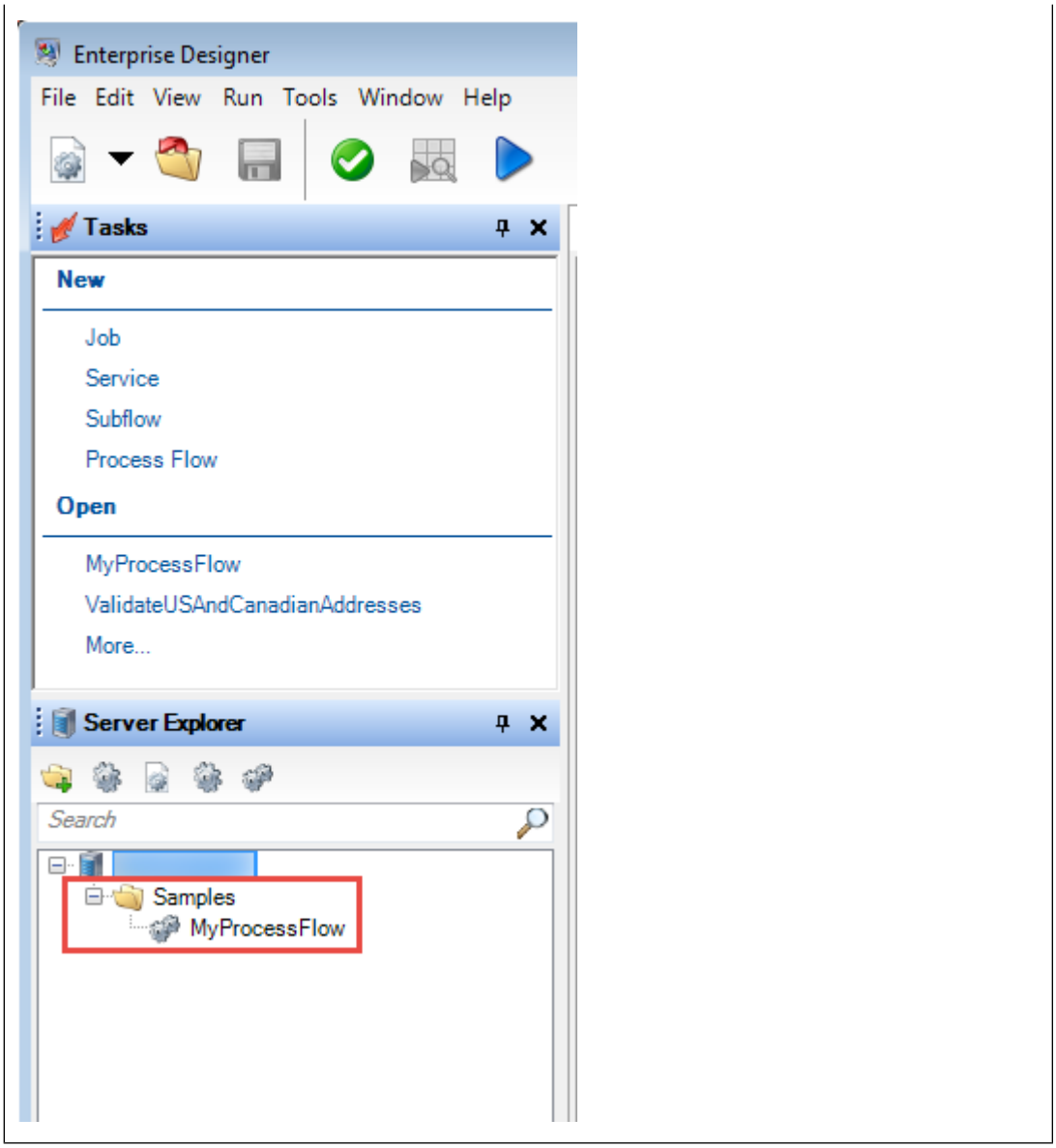
Required	Argument	Description
Yes	`--f` *ProcessFlowFile*	Specifies the process flow file (the `.pf` file) you want to import. Relative directory paths are relative to the location where you are running the Administration Utility. You can also specify an absolute path.

Required	Argument	Description
No	`--u` *TrueOrFalse*	Specifies whether to overwrite the existing process flow if a process flow with the same name is already on the server, where *TrueOfFalse* is one of the following: **true** If there is a process flow on the server with the same name as the process flow you are importing, the process flow on the server will be overwritten. This is the default setting. **false** If there is a process flow on the server with the same name as the process flow you are importing, the process flow will not be imported.
No	`--p` *Path*	Specifies the Spectrum Enterprise Designer Server Explorer folder to import the flow into.
No	`--c` *TrueOrFalse*	Specifies whether to create the folder specified in `--p` if it does not exist. **true** Create the folder specified in `--p` if it does not exist. Default. **false** Do not create the folder specified in `--p` if it does not exist. The flow will not be imported unless the folder specified in `--p` exists.

Example

This example imports the process flow named `MyProcessFlow.pf` which is located in a subfolder named `exported` in the location where you are running the Administration Utility. The process flow will be imported into the `Samples` folder in Spectrum Enterprise Designer.

```
processflow import --f exported\MyProcessFlow.pf --p Samples
```



processflow list

The `processflow list` command lists all the process flows on the server. For each process flow, the process flow name is displayed as well as whether the process flow is exposed.

Usage

```
processflow list
```

processflow history list

The `processflow history list` command shows the execution history for a process flow.

Usage

`processflow history list --n` *ProcessFlowName* `--f` *FromDateTime* `--t` *ToDateTime*

Required	Argument	Description
Yes	`--n` *ProcessFlowName*	Specifies the name of the process flow whose status you want to get. If the process flow name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the exact process flow name you can use the `processflow list` command to get a list of the process flow names.
No	`--f` *FromDateTime*	If you want to see the history for a specific date and time range, specify the starting date and time for the range, in the format MM-dd-yyyy HH:mm:ss. For example, December 31, 2014 1:00 PM would be specified as `12-31-2014 13:00:00`. When you specify a date and time range, the history list will include process flows that started execution on or after the date you specified in the `--f` argument and before the date you specify in the `--t` argument. If you omit this argument the history will include process flows that started execution on the current date.
No	`--t` *ToDateTime*	If you want to see the history for a specific date and time range, specify the ending date and time for the range, in the format MM-dd-yyyy HH:mm:ss. For example, December 31, 2014 1:00 PM would be specified as `12-31-2014 13:00:00`. When you specify a date and time range, the history list will include process flows that started execution on or after the date you specified in the `--f` argument and before the date you specify in the `--t` argument. If you omit this argument the history will include all process flows that started execution on or after the date specified in the `--f` argument.

Example

This example gets the history of the process flow named "My Process Flow".

```
processflow history list --n "My Process Flow"
```

processflow unexpose

The `processflow unexpose` command makes a process flow unavailable for execution.

Usage

`processflow unexpose --n` *ProcessFlowName*

Required	Argument	Description
Yes	`--n` *ProcessFlowName*	Specifies the name of the process flow you want to unexpose. If the process flow name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the exact process flow name you can use the `processflow list` command to get a list of the process flow names.

Example

This example unexposes the process flow named "My Process Flow".

```
processflow unexpose --n "My Process Flow"
```

processflow version list

The `processflow version list` command displays version information for all the saved versions of the process flow, including the version number, creator, creation date, comments, and which version is the exposed version.

Usage

`processflow version list --n` *ProcessFlowName*

Required	Argument	Description
Yes	`--n` *ProcessFlowName*	Specifies the name of the processflow whose version information you want to view. If the process flow name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the exact process flow name you can use the `processflow list` command to get a list of the process flow names.

Example

This example displays the version information for the process flow named "My Process Flow".

```
processflow version list --n "My Process Flow"
```

Product Data

List

- **productdata list** on page 395

Extract

- **productdata extract list** on page 395
- **productdata extract register** on page 396
- **productdata extract unregister** on page 396

Archive

- **productdata archive list** on page 396
- **productdata archive register** on page 397
- **productdata archive unregister** on page 397

Install

- **productdata install** on page 397

Delete

- **productdata delete** on page 398

List

productdata list

The `productdata list` command shows the details for currently installed Spectrum Product Data (SPD). The results provide current information without having to access the file system. We suggest that you run this command before you run the `productdata delete` command, to make an informed decision about the data to delete.

For each product installed, these details make up the product data details description:

- Product
- Component
- Qualifier
- Vintage
- Expiration [date]
- Identifier

Usage

```
productdata list
```

Extract

productdata extract list

The `productdata extract list` command shows locations of extracted product data files based on product name. The command output shows the directory where files are located for each product. The extract location "platform" shows the default extracted data location.

Usage

```
productdata extract list
```

productdata extract register

Use the `productdata extract register` command to establish an alternate (non-default) extract location for a set of product data on the server.

Usage

`productdata extract register --p` ***product*** **--d** ***directory***

Required	Argument	Description
Yes	`--p` ***product***	Specifies the product that will be referenced by this extract. The most common value is "Platform". When placing product data in another location, you might use an alternate value such as "Spectrum Global Addressing."
No	`--d` ***directory***	Defines the directory that will house the extract for the specified product. The default location is `/server/ref-data`.

productdata extract unregister

Use the `productdata extract unregister` to remove extracted product data from the server.

Usage

`productdata extract unregister --p` ***product***

Required	Argument	Description
Yes	`--p` ***product***	Specifies the product extract data that will be removed. For example, "Spectrum Global Addressing."

Archive

productdata archive list

The `productdata archive list` command shows locations where deployed product data files will be archived based on product name. The command output shows the directory where files will be archived for each product. The archive location "platform" shows the default archive location.

Usage

`productdata archive list`

productdata archive register

Use the `productdata archive register` command to establish an alternate (non-default) archive location for a set of product data on the server.

Usage

`productdata archive register --p` ***product*** **`--d`** ***directory***

Required	Argument	Description
Yes	`--p` *product*	Specifies the product that will be referenced by this archive. For example, "Platform".
No	`--d` *directory*	Defines the directory that will house the archives for the specified product. The default location is `/server/archive/ref-data`.

productdata archive unregister

Use the `productdata archive unregister` to remove archived product data from the server.

Usage

`productdata archive unregister --p` ***product***

Required	Argument	Description
Yes	`--p` *product*	Specifies the product archive that will be removed. For example, "Platform."

Install

productdata install

The `productdata install` command allows you to install Spectrum Product Data (SPD) from the command line interface (CLI).

Usage

`productdata install --f` ***fileOrDirectory*** `--w` ***waitOrReturn***

Required	Argument	Description
Yes	`--f` *fileOrDirectory*	Specifies the product file or directory location for the installation. • Specify an SPD file to install that file. • Specify a directory to install all SPD files in that directory.
No	`--w` *waitOrReturn*	Specifies the action to take during installation: wait until fully installed to return or return immediately.

Delete

productdata delete

The `productdata delete` command removes the specified Spectrum Product Data (SPD) from the Spectrum Technology Platform. Run this command on all Spectrum databases containing the SPD data that you want to remove.

Before you run this command, run the `productdata list` command to review the product, component, qualifier, and vintage details before deleting any data. The parameters to the `productdata delete` command are obtained by running the `productdata list` command.

Usage

`productdata delete --p` *productName* `--c` *productComponent* `--q` *qualifier* `--v` *dataVintage*

Required	Argument	Description
Yes	`--p` *productName*	Specifies the product using the data that will be deleted. For example, "Platform".
Yes	`--c` *productComponent*	Specifies the name of the data component to delete. For example, the product component could be "CanadaAddresses."
Yes	`--q` *qualifier*	Specifies a unique identifier for the data.
Yes	`--v` *dataVintage*	Specifies the date that the data was compiled or released. For example, a vintage could be "Q42018."

Profiles

profile run

The `profile run` command runs the specified profile.

Usage

`profile run --r` *profileRunId* `--w` *waitForComplete*

Required	Argument	Description
Yes	`--n` *profileName*	Specify the name of the profile that you want to run. If the profile name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the profile name, you can use the `profile list` command to get a list of the profile names.
No	`--w` *waitForComplete*	If `True`, the command waits for profile to complete in a synchronous mode for displaying status. **Note:** If unspecified, default is `False`.

Example

This example runs the profile "Scorecard_Demo" and it doesn't wait for the run to complete to give the status.

```
profile run --profileName Scorecard_Demo
```

profile cancel

The `profile cancel` command cancels the specified profile run.

Usage

`profile cancel --n` *profileName*

Required	Argument	Description
Yes	`--n` *profileName*	Specify the name of the profile run you want to cancel. If the profile name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the profile name, you can use the `profile list` command to get a list of the profile names.

Example

This example cancels running of the profile "Profile_CLI".

```
profile cancel --profileName Profile_CLI
```

profile delete

The `profile delete` command deletes the specified profile.

Usage

`profile delete --n` *profileName*

Required	Argument	Description
Yes	`--n` *profileName*	Specify the name of the profile you want to delete. If the profile name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the profile name, you can use the `profile list` command to get a list of the profile names.

Example

This example deletes the profile "Profile_CLI".

```
profile delete --profileName Profile_CLI
```

profile export

The `profile export` command exports reports of the specified profile to the given directory.

Usage

`profile export --n` *profileName* `--r` *profileRunId* `--t` *type* `--o` *OutputDirectory*

Required	Argument	Description
Yes	`--n` *profileName*	Specify the name of the profile you want to export. If the profile name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the profile name, you can use the `profile list` command to get a list of the names.
No	`--r` *profileRunId*	Specify the ID of the profile you want to export. If you do not provide the ID, the system fetches the ID of the latest run for the specified profile.
No	`--t` *type*	Specify the report format for profile export. Reports can be generated as PDF or an Excel file. **Note:** If unspecified, an Excel report is generated.
No	`--o` *OutputDirectory*	Specifies the directory to which you want to export the profile. The path you specify here is relative to the directory where you are running the **Administration Utility**. **Note:** If you omit this argument, the connection is exported to the directory containing the **Administration Utility**.

Example

This example exports the profile named "Scorecard" to a folder named `exported` which is a subfolder in the location where you have installed the Administration Utility. The report is in the form of a PDF document.

```
profile export --profileName Scorecard --type pdf --o exported
```

profile update

The `profile update` command updates an existing profile. Only profiles created by files on your machine or on server can be updated through this command.

Usage

`profile update --n` *profileName* `--d` *description* `--t` *profileOn* `--f` *fileName*

Note: To see a list of parameters, type `help resourceconnection import`.

Required	Argument	Description
Yes	`--n`*profileName*	Specify the name of the profile that you need to update. If the profile name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the profile name, you can use the `profile list` command to get a list of the profile names.
No	`--d`*description*	Specify the description of the profile to be updated.
True	`--t` *profileOn*	Specify the source on which the profile has been created. **Note:** Update is supported for file profiling only.
True	`--f` *fileName*	Specify the name of the file that is to be updated in profile. **Note:** The file should be in the same directory where other files of the profile are placed.

Example

This example updates the file "Scorecard_CLI.txt" in the profile "Scorecard".

```
profile update --n Scorecard --d "Running from CLI"
--profileOn Connection --f "Scorecard_CLI.txt"
```

profile status

The `profile status` command provides you the status of the profile.

Usage

`profile status --n` *profileName* `--r` *profileRunId*

Note: To see a list of parameters, type `help resourceconnection import`.

Required	Argument	Description
Yes	`--n`*profileName*	Specify the name of the profile whose status you want to view. If the profile name contains spaces, enclose the name in quotes. **Tip:** If you are unsure of the profile name, you can use the `profile list` command to get a list of the profile names.

Required	Argument	Description
No	`--r`*profileRunId*	Specify the ID of the profile run for which you want to view the status. If you do not provide the ID, the system fetches the ID of the latest run for the specified profile.

Example

This example gives the status of the profile name "Profile_CLI" for its run id "92".

```
profile status --profileName Profile_CLI --profileRunId 92
```

profile list

The `profile list` command returns a list of all the profiles created.

Usage

```
profile list
```

Note: To see a list of parameters, type `help resourceconnection import`.

Required	Argument	Description
No	`--n`*jsonFormat*	Specifies that the list of profiles are returned in JSON format.

Example

This example returns a list of all the profiles defined in Spectrum Technology Platform.

```
profile list
```

Roles

role create

The `role create` command creates a new role with the permissions defined in a JSON file.

Usage

`role create --r` *RoleName* `--f` *JSONFile*

Required	Argument	Description
Yes	`--r` *RoleName*	The name of the new role.
Yes	`--f` *JSONFile*	The path to a JSON file containing the definition for the new role.

Role File Format

The easiest way to create a role definition file is to use the `role permission export` command to generate a file of an existing role, then modify it. In this file, permissions are grouped as they are in the Spectrum Management Console list of permissions. For each secured entity, you can specify the permission for EXECUTE, DELETE, CREATE, MODIFY, and VIEW. The valid values are:

true

Grants the permission.

false

Does not grant the permission.

null

A null value indicates a permission that does not apply to the secured entity.

The following example creates a new role named `MyNewRole`. This role has permissions for the permission group `Matching`. The permissions are `Open Parser Cultures`, `Open Parser Domains`, and `OpenParser Tables`.

```
{
  "name" : "MyNewRole",
  "userNames" : [ ],
  "groups" : [ {
    "name" : "Matching",
    "permissions" : [ {
```

```
      "name" : "Open Parser Cultures",
      "permissions" : {
        "EXECUTE" : "",
        "DELETE" : "",
        "CREATE" : "",
        "MODIFY" : "true",
        "VIEW" : "true"
      }
    }, {
      "name" : "Open Parser Domains",
      "permissions" : {
        "EXECUTE" : "",
        "DELETE" : "",
        "CREATE" : "",
        "MODIFY" : "false",
        "VIEW" : "false"
      }
    }, {
      "name" : "OpenParser Tables",
      "permissions" : {
        "EXECUTE" : "",
        "DELETE" : "false",
        "MODIFY" : "false",
        "CREATE" : "false",
        "VIEW" : "false"
      }
    } ]
  } ],
}
```

Example

This example creates a new role named SalesAnalyst and uses a role definition in the file `c:\roles\SalesAnalyst.json`.

```
role create --r SalesAnalyst --f C:\roles\SalesAnalyst.json
```

role delete

The `role delete` command deletes a role.

Usage

`role delete --r` *RoleName*

Required	Argument	Description
Yes	`--r` *RoleName*	Specifies the name of the role to delete.

Example

This example deletes the role named SalesAnalyst.

```
role delete --r SalesAnalyst
```

role export

The `role export` command exports all role definitions to a JSON file named `roles.json`. This file is used as input to the `role import` command.

Usage

`role export --o` *Folder*

Required	Argument	Description
No	`--o` *Folder*	Specifies the name of the folder to which you want to export the roles. If you specify a relative path the path is relative to the location of the Administration Utility. If you do not specify a path the roles are exported to the folder where the Administration Utility is located.

Example

This example exports the roles to the RoleExports folder.

```
role export --o RoleExports
```

Note: You cannot export role permissions to directories whose names start with "\n" or "\t" are not allowed. Those character sequences are recognized as next line and tab characters, respectively. You can use forward slashes as a workaround.

Related reference

role import on page 406

role import

The `role import` command imports role definitions, and their associated permissions, from the JSON file `roles.json`, which is defined using the `role export` command.

Usage

```
role import --f File
```

Required	Argument	Description
Yes	`--f` *File*	Specifies the name of the file containing the roles and permissions. If the role does not exist, it will be created with permissions from the file. If the role exists, it will be updated. If you do not specify a path, the roles are imported from the location of the Administration Utility installation.

Example

This example imports roles to the RoleImports file.

```
role import --f RoleImports
```

Related reference

role export on page 406

role list

The `role list` command lists the names of the all the roles on the system.

Usage

```
role list
```

role export

The `role export` command exports all role definitions to a JSON file named `roles.json`. This file is used as input to the `role import` command.

Usage

```
role export --o Folder
```

Required	Argument	Description
No	`--o` *Folder*	Specifies the name of the folder to which you want to export the roles. If you specify a relative path the path is relative to the location of the Administration Utility. If you do not specify a path the roles are exported to the folder where the Administration Utility is located.

Example

This example exports the roles to the RoleExports folder.

```
role export --o RoleExports
```

Note: You cannot export role permissions to directories whose names start with "\n" or "\t" are not allowed. Those character sequences are recognized as next line and tab characters, respectively. You can use forward slashes as a workaround.

Related reference

role import on page 406

role permission import

The `role permission import` command modifies an existing role by importing permission settings from a JSON file.

Usage

`role permission import --r` *RoleName* `--f` *PermissionsFile*

Required	Argument	Description
Yes	`--r` *RoleName*	Specifies the name of the role you want to modify.
Yes	`--f` *PermissionsFile*	Specifies the name of the JSON file that contains the permissions you want to add to the role. If you specify a relative path the path is relative to the location of the Administration Utility.

Example

This example modifies the role named SalesAnayst to have the permissions defined in `c:\roles\permissions.json`.

```
role permission import --r SalesAnalyst --f
C:\roles\permissions.json
```

Scorecard

scorecard list

The `scorecard list` command lists all the scorecards created in the system.

Usage

```
scorecard list
```

scorecard evaluate

The `scorecard evaluate` command runs the specified scorecard again. It updates the scorecard, provided the path of the input file is the same as that you used while configuring the scorecard.

Usage

`scorecard evaluate --n` *scorecardName* `--f` *fileName*

Required	Argument	Description
Yes	`--n` *scorecardName*	The name of the scorecard.
No	`--w` *waitForComplete*	If `True`, the command waits for the scorecard run to complete in a synchronous mode for displaying status. **Note:** If unspecified, default is `False`.
No	`--d` *description*	Description of the scorecard.
No	`--f` *fileName*	Name of the file for updating the scorecard. The file should be placed at the same location where it was during scorecard configuration.

Example 1

This example evaluates the scorecard named *account1_Scorecard* with the input file name *account1*.

```
scorecard evaluate --n account1_Scorecard --f
c:/scorecard/account1.txt
```

Example 2

This example evaluates the scorecard named *account1_Scorecard* with the input file name *account1*. It will wait for the scorecard run to complete for giving the status.

```
scorecard evaluate --n account1_Scorecard --f
c:/scorecard/account1.txt --w True
```

scorecard status

The `scorecard status` command gives the status of the specified scorecard.

Usage

`scorecard status --n` *account1_Scorecard*

Required	Argument	Description
Yes	`--n` *scorecardName*	The name of the scorecard.
No	`--r` *scorecardRunId*	The runId for which you want the status. **Note:** If you don't provide the run Id, the status of the most recent run is displayed.

Example

This example will show the status of the scorecard named *account1_Scorecard* for runId *4*.

```
scorecard status --n account1_Scorecard --r 4
```

Related reference

role export on page 406

scorecard statistics

The `scorecard statistics` command gives the statistics related to a specified scorecard.

Usage

`scorecard statistics --n` *account1_Scorecard*

Required	Argument	Description
Yes	`--n` *scorecardName*	The name of the scorecard.
No	`--r` *scorecardRunId*	The runId for which you want the statistics. **Note:** If you don't provide the run Id, the statistics related to the most recent run is displayed.

Example

This example shows the statistics, such as rule name, KPI, and records processed for run Id 4 of a scorecard named *account1_Scorecard*.

```
scorecard statistics --n account1_Scorecard --r 4
```

The result displayed will be similar to this:

```
[{
"rulesStatistics": [{
  "totalRecords": 100,
  "ruleName": "account1_Accuracy",
  "validRecords": 100,
  "malformedRecords": 0
}],
"kpiName": "Accuracy",
"totalScore": 100,
"thresholdType": "Good"
}]
```

Related reference

role export on page 406

scorecard trends

The `scorecard trends` command gives the trend for a specified number of scorecard runs or for a specified date range.

Usage

- To view trends based on recent number of runs: `scorecard trends --n` *scorecardName* `--r` *recentRunNumber*

- To view trends on the basis of date and time: `scorecard trends --n` *scorecardName* `--v` *dateTime* `--s` *startDate* `--e` *endDate*

Required	Argument	Description
Yes	`--n` *scorecardName*	The name of the scorecard.
No	`--r` *recentRunNumber*	Recent number of runs for which you want to view the trend. **Note:** Default is 5.
No	`--v` *dateTime*	Date and time for which you want to view the trends.
	`--s` *startDate*	Start date for the scorecard trend. Date format is: *dd-MMM-yy*. **Note:** This required if argument `--v` is used.
	`--e` *endDate*	End date for the scorecard trend. Date format is: *dd-MMM-yy*. **Note:** This required if argument `--v` is used.

Example 1

This example shows trend for *10* recent runs of the scorecard named: *account1_Scorecard*.

```
scorecard trends --n account1_Scorecard --r 10
```

Example 2

This example shows the trends of scorecard runs from Jan 26, 2020 to Feb 20, 2020 for scorecard named: *account1_Scorecard*.

```
scorecard trends --n account1_Scorecard --v --s 26-Jan-20 --e
20-Feb-20
```

Related reference

role export on page 406

scorecard delete

The `scorecard delete` command deletes a scorecard.

Usage

`scorecard delete --n` *scorecardName*

Required	Argument	Description
Yes	`--n` *scorecardName*	Specifies the name of the scorecard to delete.

Example

This example deletes the scorecard named *account1_Scorecard*.

```
scorecard delete --n account1_Scorecard
```

scorecard cancel

The `scorecard cancel` command cancels a scorecard.

Usage

`scorecard cancel --n` *scorecardName*

Required	Argument	Description
Yes	`--n` *scorecardName*	Specifies the name of the scorecard to cancel.

Example

This example cancels the scorecard named *account1_Scorecard*.

```
scorecard cancel --n account1_Scorecard
```

Search Indexes

index delete

The `index delete` command deletes a Spectrum Advanced Matching search index.

Usage

```
index delete --d Name
```

Required	Argument	Description
Yes	`--d` *Name*	Specifies the name of the search index you want to delete.

Example

This example deletes a search index named "CustomerIndex".

```
index delete --d CustomerIndex
```

index list

The `index list` command returns a list of all Spectrum Advanced Matching search indexes in a tabular format. The details include index name, index type, and the number of records. Indexes can be of two types: *Legacy* or *Clustered*. You can back up and restore clustered indexes. You can export both legacy and clustered indexes to `*.txt` files using the export utility.

You can also write the index list to a CSV (`.csv`) file at any specified location.

Usage

```
index list --c checkSchema --f filePath
```

Required	Argument	Description
No	`--f` *filePath*	File path to write index list. **Note:** The output is a comma delimited text file.
No	`--c` *checkSchema*	Compares repository schema with Elasticsearch schema.

Example 1

This example writes the search index list to the file *listOutput.csv* at the location: *c:/exportLocation*.

```
index list --f c:/exportLocation/listOutput.csv
```

Example 2

This example lists all search indexes.

```
index list
```

Example 3

This example checks the schema for all the search indexes and validates the schema to return the output as True or False, and writes the output to the file *listOutput.csv* at the location: *c:/exportLocation*.

```
index list --c true --f c:/exportLocation/listOutput.csv
```

index compare

The `index compare` command compares the search index fields of repository and Elasticsearch.

Usage

`index compare --i` *indexName* `--f` *filePath*

Required	Argument	Description
Yes	`--i` *indexName*	Index name to compare between repository and Elasticsearch configuration.
Yes	`--f` *filePath*	File path to write fields of index.

Example 1

This example compares an index named "CustomerIndex" and writes the search index fields to the file *listOutput.csv* at the location: *c:/exportLocation*.

```
index compare --i CustomerIndex --f
c:/exportLocation/listOutput.csv
```

index export cancel

The `index export cancel` command cancels the search index export. The data exported till you typed the cancel command resides at the specified output location.

Usage

```
index export cancel --i Export_Id
```

Required	Argument	Description
Yes	`--i` *Export ID*	ID of the export operation to be canceled. **Note:** To find the export ID, use the index export progress command.

Example

This example cancels an export operation with this id: *Export_ID*

```
index export cancel --i Export_ID
```

index export progress

The `index export progress` command displays the status of the search indexes currently being exported. The details include, Export ID, Index name, total records, records exported, and export location.

You can also write the status of all the indexes being exported to a file.

Usage

`index export progress --f` *filePath*

Required	Argument	Description
No	`--f` *filePath*	File path to write the status of all the indexes being exported. **Note:** The output is a comma delimited text file.

Example 1

This example lists the status of all the search indexes that are currently being exported.

```
index export progress
```

Example 2

This example writes the status of all the indexes being exported to the file *exportProgress.csv* at the location: *c:/exportLocation*.

```
index export progress --f c:/exportLocation/exportProgress.csv
```

index export start

The `index export start` command exports the search index to a desired output location in `*.zip` file format. The `*.zip` file contains a `*.txt` file with pipe delimiters and double quotes as text qualifiers. The output file name corresponds to the name of the search index, followed by the time stamp.

- For **Legacy** indexes, this command exports only the fields marked as **Store** when the index is created.
- For **Clustered** indexes, this command exports all index fields.
- The line break type is `CRLF` for Windows and `LF` for non-Windows exported files.

Usage

```
index export start --i indexName --o outputLocation --d Delimiter --q
TextQualifier
```

Required	Argument	Description
Yes	`--i` *indexName*	Specifies the name of the search index to export.
Yes	`--o` *outputLocation*	Specifies the output location for the exported index. If you do not specify an output location, you will see a message reminding you to do this.
No	`--d` *Delimiter*	File Delimiter of the output file. By default, it is set to pipe. **Note:** Delimiter values for Space and Tab are "\ " and "\t" respectively.
No	`--q` *TextQualifier*	Text Qualifier of the output file. By default, it is set to double quotes. **Note:** You don't need to specify the qualifier value if you want to set it to double quotes, as it is the default value.

Example 1

This example exports a search index named "CustomerIndex" to output location "pbIndexExports."

```
index export start --i CustomerIndex --o c:/pbIndexExports
```

Example 2

This example exports the search index *sample_index* to an output location `/home/exportLocation` with delimiter as dollar sign and qualifier as curl sign.

```
index export start --i sample_index --o /home/exportLocation
--d $ --q ^
```

index snapshot create

The `index snapshot create` command creates a snapshot of a search index. For a snapshot to be created successfully, the entire data set in the index needs to be valid. Any missing primary shard will result in failure of snapshot creation. Once a snapshot is created, its subsequent back up takes much less time since it requires incremental addition and deletion of data.

Snapshot creation is not affected by any operation being performed on the search index in parallel. However, only the records present in the index at that particular time gets recorded in the snapshot.

To view the status of the snapshot you created, use the command `index snapshot list`. For more information see **index snapshot list** on page 419.

> **Note:** You need to create an index snapshot repository before you can create snapshot of a search index. For more information, see **index snapshot repository** on page 420.

Usage

```
index snapshot create
```

Required	Argument	Description
Yes	`--i` *Index name*	Name of the search index in the snapshot to be created.
Yes	`--s` *Snapshot name*	Name of the snapshot to be created. **Note:** The name needs to be unique. **Note:** It will be in lowercase irrespective of the casing you use in the command.

Example

This example creates a new snapshot "my_snapshot" of search index "customer_index".

```
index snapshot create --i customer_index --s my_snapshot
```

index snapshot list

The `index snapshot list` command returns a list of all search index snapshots. The details displayed are:

- Name of the snapshot
- Name of the search index
- If the snapshot was created successfully
- Reason of failure, if the snapshot was not created successfully
- The start time, end time and total time taken in creating the snapshot
- Total shards in the snapshot and successful and failed shards, if any

You can also write the search index snapshot list to a file.

Usage

`index snapshot list --f` *filePath*

Required	Argument	Description
No	`--f` *filePath*	File path to write the search index snapshot list. **Note:** The output is a comma delimited text file.

Example 1

This example lists all the search index snapshots.

```
index snapshot list
```

Example 2

This example writes the search index snapshot list to the file *snapshotList.csv* at the location: *c:/exportLocation*.

```
index snapshot list --f c:/exportLocation/snapshotList.csv
```

index snapshot delete

The `index snapshot delete` command deletes a search index snapshot.

Usage

`index snapshot delete`

Required	Argument	Description
Yes	`--i` *Index name*	Name of the search index, the snapshot of which is to be deleted.

Required	Argument	Description
Yes	`--s` *Snapshot name*	Name of the snapshot to be deleted.

Example

This example deletes the snapshot "my_snapshot" of search index "customer_index".

```
index snapshot delete --i customer_index --s my_snapshot
```

index snapshot repository

The `index snapshot repository` command sets the search index snapshot repository shared file system path. You can set any number of repositories. But, the Search Index Engine always uses the currently set file path for backing up and restoring data.

To use the index snapshot repository for search index (in CLI), you need to first specify the repository path, `path.repo` at the location given below and restart the server. In case of cluster set-up this modification needs to be done at all the nodes. This path is used while creating the repository *SpectrumDirectory*`\index\elasticsearch.template`.

For example, the path can be:

- `path.repo: ["/mount/backups"]`
- `path.repo: ["C:/SIbackups"]`

Examples of repositories created using the above path are:

- `index snapshot repository --p /mount/backups/index_customer`
- `index snapshot repository --p C:/SIbackups/index_customer`

Usage

```
index snapshot repository
```

Required	Argument	Description
Yes	`--p` *Name*	Specifies the path where the index snapshot repository is to be set. **Note:** It is mandatory to use a shared file system path.
No	`--o` *Overwrite*	Overwrites the already set path with the new one.

Example

This example sets a new index snapshot repository at the path "c:\\MyIndexRepository".

```
index snapshot repository --p c:\\MyIndexRepository
```

index snapshot restore

The `index snapshot restore` command restores a search index snapshot.

Note: No operation is allowed on the search index while you are restoring its snapshot.

To view the status of snapshot restore use the command `index restore list`. For more information, see **index snapshot list** on page 419

Usage

```
index snapshot restore
```

Required	Argument	Description
Yes	`--i` *Index name*	Name of the search index, the snapshot of which is being restored.
Yes	`--s` *Snapshot name*	Name of the snapshot to be restored.

Example

This example restores the snapshot "my_snapshot" of search index "customer_index".

```
index snapshot restore --i customer_index --s my_snapshot
```

index restore list

The `index restore list` command returns a list of all the restored index snapshots of Spectrum Advanced Matching. The details, in a tabular format, includes the name of the index and its restore status. It also gives the shard-wise restoration details, such as the total time taken in restoring each shard, the status, and restoration description.

You can also write the list of all the restored index snapshots to a file.

Usage

```
index restore list --f filePath
```

Required	Argument	Description
No	`--f` *filePath*	File path to write the list of all the restored index snapshots. **Note:** The output is a comma delimited text file.

Example 1

This example lists all the restored search index snapshots.

```
index restore list
```

Example 2

This example writes the status of snapshot restores to the file *restoreList.csv* at the location: *c:/exportLocation*.

```
index restore list --f c:/exportLocation/restoreList.csv
```

Services

service list

The `service list` command lists the services that are exposed on your server. Services that are not exposed will not be shown in the list.

Note: You can expose services using the `dataflow expose` command.

Usage

```
service list
```

service loglevel list

The `service loglevel list` command lists the level of detail included in the log for each service. The log levels are:

Default	The service uses the system default logging level. You can set the system log level by using the `systemloglevel set` command.
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.

Usage

```
service loglevel list
```

service loglevel set

The `service loglevel set` command specifies the level of detail included in the service log.

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.

Usage

`service loglevel set --s` *ServiceName* `--l` *LogLevel*

Required	Argument	Description
Yes	`--s` *ServiceName*	Specifies the name of the service whose logging level you want to set.
Yes	`--l` *LogLevel*	Specifies the logging level for the service, where *LogLevel* is one of the following: **Default** The service uses the system default logging level. You can set the system log level by using the `systemloglevel set` command. **Disabled** No event logging enabled. **Fatal** Minimal logging. Only fatal errors are logged. Fatal errors are those that make the system unusable.

Required	Argument	Description	
		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 most intensive logging level can affect system performance. Therefore, you should select the least intensive setting that meets your particular logging requirements.

Example

This example sets the logging level for ValidateAddress to Warn:

```
service loglevel set --s ValidateAddress --l Warn
```

service option list

The `service option list` command lists the options in effect for a service. For a description of each service's options and their values, see one of the following: *API Guide*, *REST Web Services Guide*, or *SOAP Web Services Guide*.

Usage

`service option list --s` *ServiceName*

Required	Argument	Description
Yes	`--s` *ServiceName*	Specifies the name of the service whose options you want to view. Service names are case sensitive.

Example

This example lists the options in effect for the ValidateAddress service:

```
service option list --s ValidateAddress
```

service option set

The `service option set` command specifies a default setting for a service option.

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

Usage

`service option set --s` *ServiceName* `--o` *OptionName* `--v` *Value*

Required	Argument	Description
Yes	`--s` *ServiceName*	Specifies the name of the service for which you want to set an option. Service names are case sensitive.
Yes	`--o` *OptionName*	The name of the option you want to set. For a description of each service's options and their values, see one of the following: *API Guide*, *REST Web Services Guide*, or *SOAP Web Services Guide*.

Required	Argument	Description
Yes	`--v` *Value*	The value you want to set for the option. Separate multiple values with a comma. For a description of each service's options and their values, see one of the following: *API Guide*, *REST Web Services Guide*, or *SOAP Web Services Guide*.

Example

This example sets the MaximumResults option for the ValidateAddress service to 15:

```
service option set --s ValidateAddress --o MaximumResults --v 15
```

Spectrum Databases

Spectrum Enterprise Geocoding for Global Databases

- **egmglobaldb create sample file** on page 429
- **egmglobaldb delete** on page 430
- **egmglobaldb import** on page 430
- **egmglobaldb export** on page 430
- **egmglobaldb get** on page 431
- **egmglobaldb list** on page 431
- **egmglobaldb memory set** on page 432
- **egmglobaldb poolsize set** on page 433

Spectrum Enterprise Geocoding for US Databases

- **egmusadb add** on page 433
- **egmusadb delete** on page 434
- **egmusadb import** on page 434
- **egmusadb export** on page 434
- **egmusadb get** on page 435
- **egmusadb list** on page 436
- **egmusadb memory set** on page 436
- **egmusadb poolsize set** on page 437

Spectrum Enterprise Geocoding for World Databases

- **egmworlddb create_sample_file** on page 438
- **egmworlddb delete** on page 439
- **egmworlddb import** on page 439
- **egmworlddb export** on page 439
- **egmworlddb get** on page 440
- **egmworldb list** on page 440
- **egmworlddb memory set** on page 441
- **egmworlddb poolsize set** on page 442

Spectrum Enterprise Tax Databases

- **geotaxdb delete** on page 442
- **geotaxdb import** on page 443
- **geotaxdb export** on page 443
- **geotaxdb get** on page 444
- **geotaxdb list** on page 444
- **geotaxdb memory set** on page 445
- **geotaxdb poolsize set** on page 446

Spectrum Global Addressing Databases

- **gamdb create** on page 446
- **gamdb delete** on page 448
- **gamdb export** on page 448
- **gamdb get info** on page 449
- **gamdb import** on page 451
- **gamdb listdatasets** on page 451
- **gamdb listdbresources** on page 451
- **gamdb memory set** on page 452
- **gamdb modify** on page 453
- **gamdb poolsize set** on page 454

Spectrum Global Geocoding Databases

- **globalgeocodedb create sample file** on page 455
- **globalgeocodedb delete** on page 456
- **globalgeocodedb import** on page 456
- **globalgeocodedb export** on page 457
- **globalgeocodedb get** on page 458
- **globalgeocodedb list** on page 459

- **globalgeocodedb memory set** on page 459
- **globalgeocodedb poolsize set** on page 460

Spectrum Spatial and Routing Databases

- **limrepo export** on page 461
- **limrepo import** on page 462
- **limrepo mwsimport** on page 465
- **ermdb list** on page 466
- **ermdb get** on page 467
- **ermdb add** on page 467
- **ermdb delete** on page 468
- **ermdb memory set** on page 469
- **ermdb modify** on page 470
- **ermdb template** on page 471
- **ermdb import** on page 473
- **ermdb export** on page 474
- **erm getpointdata** on page 475
- **erm getsegmentdata** on page 476
- **erm createpointupdate** on page 477
- **erm resetpointupdate** on page 479
- **erm createsegmentupdate** on page 480
- **erm resetsegmentupdate** on page 482
- **erm getsegmentupdates** on page 483
- **erm createroadtypeupdate** on page 484
- **erm resetroadtypeupdate** on page 486
- **erm getroadtypeupdates** on page 487
- **erm getallupdates** on page 488
- **erm resetallupdates** on page 489

Spectrum Universal Address Databases

- **uamdb create** on page 489
- **uamdb modify** on page 491
- **uamdb delete** on page 492
- **uamdb memory set** on page 492
- **uamdb import** on page 493
- **uamdb export** on page 494
- **uamdb get resource info** on page 494
- **uamdb list** on page 495
- **uamdb listdatasets** on page 496

- **uamdbglobalmultipath create_sample_file** on page 496
- **uamdb poolsize set** on page 497

Spectrum Enterprise Geocoding for Global Databases

egmglobaldb create sample file

The `egmglobaldb create_sample_file` command creates sample JSON file of single and double database resource. These generated files can be used as reference for providing configurations for creation of database resource. It creates *egmGlobalSingleDictDbResource.txt* and *egmGlobalDoubleDictDbResource.txt* JSON files in the current directory or at specified folder location.

Usage

`egmglobaldb create_sample_file` ***outputpath***

Required	Argument	Description
No	*outputpath*	All the sample database resources JSON files will be created at provided output path else all will be exported to current folder.

Example

This example creates the sample global database resources JSON files to current folder. The second example will export all the database resources to C:\OutputFolder.

```
egmglobaldb create_sample_file
```

Sample JSON for database resource file

```
egmglobaldb create_sample_file C:\OutputFolder
```

```
[{"product":"InternationalGeocoder GLOBAL", "module":"igeocode-global",
"name":"$$DATABASE_NAME$$",
"maxActive":4,
"properties":{"COUNTRY_CODE1":"$$COUNTRY_CODE1$$",
"$$COUNTRY_CODE1$$_DICTIONARY_PATH1":"$$DICTIONARY_PATH1$$",
"COUNTRY_COUNT":"1",
"$$COUNTRY_CODE1$$_DICTIONARY_PATH_NAME1":"$$DICTIONARY_PATH_NAME1$$"}}]
```

egmglobaldb delete

The `egmglobaldb delete` command deletes a configured Spectrum Enterprise Geocoding Global database.

Usage

`egmglobaldb delete --n` *Name*

Required	Argument	Description
Yes	`--n` *Name*	Specifies the name of the database.

Example

This example deletes the global database from the global module.

```
egmgbrdb delete --n Global
```

egmglobaldb import

The `egmglobaldb import` command imports a Spectrum Enterprise Geocoding Global database property file. This configures the Global database resources on the current system.

Usage

`egmglobaldb import --f` *File*

Required	Argument	Description
Yes	`--f` *File*	Specifies the JSON-formatted database property file. This file is mandatory.

Example

This example creates a Global database resource as defined by the configuration provided in the `egmGlobalSingleDictDbResource.txt` JSON-formatted file.

```
egmglobaldb import --f egmGlobalSingleDictDbResource.txt
```

egmglobaldb export

The `egmglobaldb export` command exports all of the Global database resource information to a database properties file, `EgmGlobalDbResource.txt`, at the specified location. If the location for the output file is not provided, the `EgmGlobalDbResource.txt` file is written to the current folder. The database properties file can subsequently be used with the `egmglobaldb import` command to configure the databases on another system.

Usage

`egmglobaldb export --o` ***outputpath***

Required	Argument	Description
No	`--o` *outputpath*	All the database resources will be exported to the provided output path. If path is not specified, all resources will be exported to current folder. The exported EgmGlobalDbResource.txt file JSON-formatted output file contains the database properties information.

Example

This example exports the database resource information to the designated location.

```
egmglobaldb export --o C:\DBs\
```

egmglobaldb get

The `egmglobaldb get` command returns information about a Spectrum Global Enterprise Geocoding database.

Usage

`egmglobaldb get --n` ***Name***

Required	Argument	Description
Yes	`--n` *Name*	Specifies the name of the database. This file is mandatory.

Example

This example displays all the information for the configured Global database resource.

```
egmglobaldb get --n Global
```

egmglobaldb list

The `egmglobaldb list` command displays all the configured Spectrum Enterprise Geocoding Global databases and their pool sizes.

Usage

`egmglobaldb list` This command does not have any properties.

Example

This example lists the Spectrum Enterprise Geocoding Global database and the pool size.

```
egmglobaldb list
```

egmglobaldb memory set

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 272.

The `egmglobaldb memory set` command defines the memory size for Spectrum Enterprise Geocoding for Global databases. You must have Spectrum Global Geocoding - Global installed to use this command. The fields for defining minimum and maximum memory values can be empty. If a value is empty, that value will not be specified on the command line when starting the component, as if no value were explicitly defined. If no value is specified, or if a value is 0, the property will not be passed to the Command Line Interface.

Usage

`egmglobaldb memory set --name` *database_name* `--mn` *minimum_memory_size* `--mx` *maximum_memory_size*

Note: To see a list of parameters, type `help egmglobaldb memory set`.

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 `egmglobaldb list` command.
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 sets the database memory sizes for a second quarter database.

```
egmglobaldb memory set --name egmglobalq2 --mn 1200 --mx 65536
```

egmglobaldb poolsize set

The `egmglobaldb poolsize set` command sets the pool size for a configured global database resource. The pool size is the maximum number of concurrent requests allowed to a database.

Usage

`egmglobaldb poolsize set --n` *Name* `--s` *Poolsize*

Required	Argument	Description
Yes	`--n` *Name*	Specifies the name of the database.
No	`--s` *Poolsize*	Sets the pool size (specified as an integer value) of the database. The default is 4.

Example

This example sets the poolsize of an already configured Global database resource to 10.

```
egmglobaldb poolsize set --n global --s 10
```

Spectrum Enterprise Geocoding for US Databases

egmusadb add

The `egmusadb add` command creates a new US Spectrum Enterprise Geocoding database resource on the server. You must have Spectrum Enterprise Geocoding US installed to use this command.

Usage

`egmusadb add --f` *file* `--mn` *minimum_memory_size* `--mx` *maximum_memory_size*

Required	Argument	Description
Yes	`--f` *file*	Specifies the directory and name of the database resource to be added.
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.

egmusadb delete

The `egmusadb delete` command deletes a configured Spectrum Enterprise Geocoding US database.

Usage

`egmusadb delete --n` *name*

Required	Argument	Description
Yes	`--n` *name*	Specifies the name of the database.

Example

This example deletes a database named "EGM_CENTRUS_POINTS".

```
egmusadb delete --n EGM_CENTRUS_POINTS
```

egmusadb import

The `egmusadb import` command imports a Spectrum Enterprise Geocoding US database property file created by the `egmusadb export` command. The `egmusadb import` command configures the US database resource on the current system.

Usage

`egmusadb import --f` *file*

Required	Argument	Description
Yes	`--f` *file*	Specifies the JSON-formatted database property file.

Example

This example will create a United States database resource as defined by the configuration provided in the `EgmDbResource.txt` JSON-formatted file.

```
egmusadb import --f EgmDbResource.txt
```

egmusadb export

The `egmusadb export` command exports all of the United States database resource information to a database property file, `EgmDbResource.txt`, at the specified location. If the location for the output file is not provided, the EgmDbResource.txt file is written to the current folder. The database property file can subsequently be used with the `egmusadb import` command to configure the databases on another system.

Usage

`egmusadb export --o` *directory*

Required	Argument	Description
No	`--o` *directory*	Specifies the output directory to store the JSON-formatted output file, EgmDbResource.txt, which contains the database properties information.

Example

This example exports the database information to the designated location.

```
egmusadb export --o C:\DBs\
```

The `EgmDbResource.txt` output file contains database property information similar to the following:

```
[{"product":"GeoStan",
"module":"geostan",
"name":"TomTomStreets",
"maxActive":4,
"properties":{"BASE_DB_PATHS":"C:/Dataset/DVDGDT",
"DataSetName":"TomTomStreets"}},
{"product":"GeoStan",
"module":"geostan",
"name":"CentrusPoints",
"maxActive":4,
"properties":{"BASE_DB_PATHS":"C:/Dataset/DVDCPoints;C:/Dataset/DVDGDT",
"DataSetName":"CentrusPoints"}}]
```

egmusadb get

The `egmusadb get` command returns information about a US Spectrum Enterprise Geocoding database.

Usage

`egmusadb get --n` *name*

Required	Argument	Description
Yes	`--n` *name*	Specifies the name of the database.

Example

This example retrieves information for a database named "EGM_CENTRUS_PTS".

```
egmusadb get --n EGM_CENTRUS_PTS
```

The returned information may be similar to the following:

```
DATABASE NAME = EGM_CENTRUS_PTS
POOL SIZE = 4
BASE_DB_PATH = C:\DBs\EGM\CENTRUS_JUL14
DataSetName = EGM_CENTRUS_PTS
```

egmusadb list

The `egmusadb list` command displays all the configured Spectrum Enterprise Geocoding US databases and their pool size.

Usage

`egmusadb list` This command does not have any properties.

Example

This example lists all the Spectrum Enterprise Geocoding US databases.

```
egmusadb list
```

The returned information may be similar to the following:

```
+----------------+---------+
|  DATABASE NAME |POOL SIZE|
+----------------+---------+
| TomTomStreets  |       4 |
| TomTomPoints   |       4 |
| NAVTEQStreets  |       4 |
| CentrusPoints  |       4 |
+----------------+---------+
```

egmusadb memory set

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 272.

The `egmusadb memory set` command defines the memory size for the Spectrum Enterprise Geocoding US database. You must have Spectrum Enterprise Geocoding US installed to use this command. The fields for defining minimum and maximum memory values can be empty. If a value is empty, that value will not be specified on the command line when starting the component, as if no value were explicitly defined. If no value is specified, or if a value is 0, the property will not be passed to the Command Line Interface.

Usage

`egmusadb memory set --name` ***database_name*** `--mn` ***minimum_memory_size*** `--mx` ***maximum_memory_size***

Note: To see a list of parameters, type `help egmusadb memory set`.

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 `egmusadb list` command.
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 sets the database memory sizes for the Spectrum Enterprise Geocoding US database.

```
egmusadb memory set --name EGMMUS --mn 1200 --mx 65536
```

egmusadb poolsize set

The `egmusadb poolsize set` command sets the pool size for a configured Spectrum Enterprise Geocoding US database. The pool size is the maximum number of concurrent requests allowed to a database.

Usage

`egmusadb poolsize set --n` ***name*** `--s` ***poolsize***

Required	Argument	Description
Yes	`--n` *name*	Specifies the name of the database.
No	`--s` *poolsize*	Sets the pool size (specified as an integer value) of the database. The default is 4.

Example

This example sets the poolsize to '3' for the "EGM_CENTRUS_PTS" database.

```
egmusadb poolsize set --n EGM_CENTRUS_PTS --s 3
```

Spectrum Enterprise Geocoding for World Databases

egmworlddb create_sample_file

The `egmworlddb create_sample_file` command creates sample JSON files of single and double database resource. These generated files can be used as reference for providing configurations for creation of database resource. It creates *egmWorldSingleDictDbResource.txt* and *egmWorldDoubleDictDbResource.txt* JSON files in the current directory or at specified folder location.

Usage

`egmworlddb create_sample_file` ***outputpath***

Required	Argument	Description
No	*outputpath*	All the sample database resources JSON files will be created at provided output path else all will be exported to current folder.

Example

This example creates the sample world database resources JSON files to current folder. The second example will export all the database resources to C:\OutputFolder.

```
egmworlddb create_sample_file
```

Sample JSON for database resource file

```
egmworlddb create_sample_file C:\OutputFolder
```

```
[{"product":"InternationalGeocoder WORLD", "module":"igeocode-world",
"name":"$$DATABASE_NAME$$",
"maxActive":4,
"properties":{"COUNTRY_CODE1":"$$COUNTRY_CODE1$$",
"$$COUNTRY_CODE1$$_DICTIONARY_PATH1":"$$DICTIONARY_PATH1$$",
"COUNTRY_COUNT":"1",
"$$COUNTRY_CODE1$$_DICTIONARY_PATH_NAME1":"$$DICTIONARY_PATH_NAME1$$"}}]
```

egmworlddb delete

The `egmworlddb delete` command deletes a configured Spectrum Enterprise Geocoding World database.

Usage

`egmworlddb delete --n` *Name*

Required	Argument	Description
Yes	`--n` *Name*	Specifies the name of the database.

Example

This example deletes the World database from the World module.

```
egmgbrdb delete --n world
```

egmworlddb import

The `egmworlddb import` command imports a Spectrum Enterprise Geocoding World database property file. This configures the World database resource on the current system.

Usage

`egmworlddb import --f` *File*

Required	Argument	Description
Yes	`--f` *File*	Specifies the JSON-formatted database property file. This file is mandatory.

Example

This example creates a World database resource as defined by the configuration provided in the `egmGlobalSingleDictDbResource.txt` JSON-formatted file.

```
egmglobaldb import --f egmWorldSingleDictDbResource.txt
```

egmworlddb export

The `egmworlddb export` command exports all of the World database resource information to a database properties file, `EgmWorldDbResource.txt`, at the specified location. If the location for the output file is not provided, the `EgmWorldDbResource.txt` file is written to the current folder. The database properties file can subsequently be used with the `egmworlddb import` command to configure the databases on another system.

Usage

`egmworlddb export --o` ***outputpath***

Required	Argument	Description
No	`--o` *outputpath*	All the database resources will be exported to the provided output path. If path is not specified, all resources will be exported to current folder. The exported EgmWorldDbResource.txt file JSON-formatted output file contains the database properties information.

Example

This example exports the database resource information to the designated location.

```
egmworlddb export --o C:\DBs\
```

egmworlddb get

The `egmworlddb get` command returns information about a Spectrum Enterprise Geocoding Global database.

Usage

`egmworlddb get --n` ***Name***

Required	Argument	Description
Yes	`--n` *Name*	Specifies the name of the database. This file is mandatory.

Example

This example displays all the information for the configured World database resource.

```
egmworlddb get --n World
```

egmworldb list

The `egmworlddb list` command displays all the configured Spectrum Enterprise Geocoding World databases and their pool sizes.

Usage

`egmworlddb list` This command does not have any properties.

Example

This example lists the Spectrum Enterprise Geocoding World database and the pool size.

```
egmworlddb list
```

egmworlddb memory set

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 272.

The `egmworlddb memory set` command defines the memory size for Spectrum Enterprise Geocoding World databases. You must have Spectrum Enterprise Geocoding World installed to use this command. The fields for defining minimum and maximum memory values can be empty. If a value is empty, that value will not be specified on the command line when starting the component, as if no value were explicitly defined. If no value is specified, or if a value is 0, the property will not be passed to the Command Line Interface.

Usage

`egmworlddb memory set --name` ***database_name*** `--mn` ***minimum_memory_size*** `--mx` ***maximum_memory_size***

Note: To see a list of parameters, type `help egmworlddb memory set`.

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	`--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 sets the database memory sizes for the Spectrum Enterprise Geocoding US World database.

```
egmworlddb memory set --name EGMUS --mn 1200 --mx 65536
```

egmworlddb poolsize set

The `egmworlddb poolsize set` command sets the pool size for a configured World database resource. The pool size is the maximum number of concurrent requests allowed to a database.

Usage

`egmworlddb poolsize set --n` *Name* `--s` *Poolsize*

Required	Argument	Description
Yes	`--n` *Name*	Specifies the name of the database.
No	`--s` *Poolsize*	Sets the pool size (specified as an integer value) of the database. The default is 4.

Example

This example sets the poolsize of an already configured Global database resource to 10.

```
egmworlddb poolsize set --n world --s 10
```

Spectrum Enterprise Tax Databases

geotaxdb delete

The `geotaxdb delete` command deletes a configured Spectrum Enterprise Tax database.

Usage

`geotaxdb delete --n` *name*

Required	Argument	Description
Yes	`--n` *name*	Specifies the name of the database.

Example

This example deletes a database named "ETM_CENTRUS_POINTS".

```
geotaxdb delete --n ETM_CENTRUS_POINTS
```

geotaxdb import

The `geotaxdb import` command imports a Spectrum Enterprise Tax database property file created by the `geotaxdb export` command. The `geotaxdb import` command configures the database resource on the current system.

Usage

`geotaxdb import --f` *file*

Required	Argument	Description
Yes	`--f` *file*	Specifies the JSON-formatted database property file.

Example

This example will create a Spectrum Enterprise Tax database resource as defined by the configuration provided in the `GeotaxDbResource.txt JSON-formatted` file.

```
geotaxdb import --f GeotaxDbResource.txt
```

geotaxdb export

The `geotaxdb export` command exports all of the Spectrum Enterprise Tax database resource information to a database property file, `GeotaxDbResource.txt`, at the specified location. If the location for the output file is not provided, the `GeotaxDbResource.txt` file is written to the current folder. The database property file can subsequently be used with the `geotaxdb import` command to configure the databases on another system.

Usage

`geotaxdb export --o` *directory*

Required	Argument	Description
No	`--o` *directory*	Specifies the output directory to store the JSON-formatted output file, GeotaxDbResource.txt, which contains the database properties information.

Example

This example exports the database information to the designated location.

```
geotaxdb export --o C:\Data\
```

The `GeotaxDbResource.txt` output file contains database property information similar to the following:

```
[ {
  "product" : "Spectrum Enterprise Tax",
  "module" : "gtx",
  "name" : "ETM_DB",
  "maxActive" : 4,
  "properties" : {
    "BASE_DB_PATH" : "C:/Datasets/DVDGTX",
    "POINTS_DB_PATH" : "C:/Datasets/DVDMLD"
  }
} ]
```

geotaxdb get

The `geotaxdb get` command returns information about a Spectrum Enterprise Tax database.

Usage

`geotaxdb get --n` *name*

Required	Argument	Description
Yes	`--n` *name*	Specifies the name of the database.

Example

This example retrieves information for a database named "ETM_6".

```
geotaxdb get --n ETM_6
```

The returned information may be similar to the following:

```
DATABASE NAME = ETM_6
POOL SIZE = 4
BASE_DB_PATH = C:/Datasets/DVDGTX
POINTS_DB_PATH = C:/Datasets/DVDMLD
```

geotaxdb list

The `geotaxdb list` command displays all the configured Spectrum Enterprise Tax databases and their pool size.

Usage

`geotaxdb list` This command does not have any properties.

Example

This example lists all the Spectrum Enterprise Tax databases.

```
geotaxdb list
```

The returned information may be similar to the following:

```
+----------------+---------+
|  DATABASE NAME |POOL SIZE|
+----------------+---------+
| TomTomStreets  |       4 |
| TomTomPoints   |       4 |
| NAVTEQStreets  |       4 |
| CentrusPoints  |       4 |
+----------------+---------+
```

geotaxdb memory set

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 272.

The `geotaxdb memory set` command defines the memory size for the Spectrum Enterprise Tax databases. You must have Spectrum Enterprise Tax installed to use this command. The fields for defining minimum and maximum memory values can be empty. If a value is empty, that value will not be specified on the command line when starting the component, as if no value were explicitly defined. If no value is specified, or if a value is 0, the property will not be passed to the Command Line Interface.

Usage

`geotaxdb memory set --name` ***database_name*** `--mn` ***minimum_memory_size*** `--mx` ***maximum_memory_size***

Note: To see a list of parameters, type `help geotaxdb memory set`.

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	`--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.

Required	Argument	Description
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 sets the database memory sizes for a database named ETM Centrus Points.

```
geotaxdb memory set --name ETM_CENTRUS_POINTS --mn 1200 --mx
65536
```

geotaxdb poolsize set

The `geotaxdb poolsize set` command sets the pool size for a configured Spectrum Enterprise Tax database. The pool size is the maximum number of concurrent requests allowed to a database.

Usage

`geotaxdb poolsize set --n` *name* `--s` *poolsize*

Required	Argument	Description
Yes	`--n` *name*	Specifies the name of the database.
No	`--s` *poolsize*	Sets the pool size (specified as an integer value) of the database. The default is 4.

Example

This example sets the poolsize to '3' for the "ETM_CENTRUS_PTS" database.

```
geotaxdb poolsize set --n ETM_CENTRUS_PTS --s 3
```

Spectrum Global Addressing Databases

gamdb create

The `gamdb create` command creates and configures Spectrum Global Addressing Management databases.

Usage

`gamdb create --n` *Name* `--d` *Dataset Name* `--v` *Dataset Vintage* `--c` *Country* `--t` *Type* `--g` *Group* `--p` *Poolsize* `--mn` *minimum_memory_size* `--mx` *maximum_memory_size*

Required	Argument	Description
Yes	`--n` *Name*	Specifies the name of the database resource to create.
Yes	`--d` *Dataset Name*	Specifies the name of the SPD dataset that is the Qualifier name.
Yes	`--v` *Dataset Vintage*	Specifies the vintage of the dataset.
No	`--c` *Country*	Specifies the three-digit ISO code for each country to include in the databases specified by the "t" option (type of SPD) where Countries is a list of three-digit ISO codes separated by semicolons. For more information about ISO codes, see your *Spectrum Technology Platform Addressing Guide*.
Yes	`--t` *Type*	Specifies the type of dataset. **GAV** Spectrum Global Address Validation dataset. **GTA** Spectrum Global Type Ahead dataset.
Yes	`--g` *Group*	Specifies the coder for Spectrum Global Address Validation. **Global** Spectrum Global Address Validation International coder. **US** Spectrum Global Address Validation US coder.
No	`--p` *Poolsize*	Specifies the maximum number of concurrent requests you want this database to handle. The default is 4.
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 creates a Spectrum Global Address Validation database for Germany named "GAV_DEU" using the database resource "GAV_EMEA" with a December 2018 vintage and the International coder. This example configures the GAV_DEU database with a pool size of 5, and memory allocation between 12200 and 65536 MB.

```
gamdb create --n GAV_DEU --d GAV_EMEA --v DEC2018 --c DEU --t
GAV --g Global --p 5 --mn 12200 --mx 65536
```

Example

This example creates a Spectrum Global Type Ahead database for Austria named "GTA_AUT using the database resource "GTA_EMEA" with a December 2018 vintage. This example configures the GTA_AUT database with a pool size of 6, and memory allocation between 12200 and 65536 MB.

```
gamdb create --n GTA_AUT --d GTA_EMEA --v DEC2018 --c AUT --t
GTA --p 6 --mn 12200 --mx 65536
```

gamdb delete

The `gamdb delete` command deletes a Spectrum Global Addressing database.

Usage

`gamdb delete --n` *Name* `--g` *Group*

Required	Argument	Description	
Yes	`--n` *Name*	Specifies the name of the database.	
Yes	`--g` *Group*	Specifies the coder for Global Address Validation.	
		Global	Global Address Validation International coder.
		US	Global Address Validation US coder.

Example

This example deletes a Global Address Validation database for Germany named "GAV_DEU". This example specifies the Global Address Validation International coder.

```
gamdb delete --n GAV_DEU --g Global
```

Example

This example deletes a Global Type Ahead database for Austria named "GTA_AUT".

```
gamdb delete --n GTA_AUT
```

gamdb export

The `gamdb export` command exports all of the Global Addressing database resource information to a database properties file, `GlobalAddressingDbResource.txt`, either at a specified location, or if the location for the output file is not provided, `GlobalAddressingDbResource.txt` is written to the current folder. The database properties file can subsequently be used with the `gamdb import` command to configure the databases on another system.

Usage

`gamdb export --o` *outputpath* `--g` *Group*

Required	Argument	Description
No	`--o` *outputpath*	The information on the Global Addressing database resources will be exported to `GlobalAddressingDbResource.txt` in the specified output directory. If the path is not provided, `GlobalAddressingDbResource.txt` will be written to the current folder.
Yes	`--g` *Group*	Specifies the Global Address Validation coder. **Global** Global Address Validation International coder. **US** Global Address Validation US coder.

Example

This example exports the Global Addressing database resource information to the designated directory. This example specifies the Global Address Validation International coder.

```
gamdb export --o C:\DBs\ --g Global
```

gamdb get info

The `gamdb get info` command returns detailed information about a Global Addressing database.

Usage

`gamdb get info --n` *Name* `--g` *Group*

Required	Argument	Description
Yes	`--n` *Name*	Specifies the name of the database.
Yes	`--g` *Group*	Specifies the coder for Global Address Validation. **Global** Global Address Validation International coder. **US** Global Address Validation US coder.

Example

This example displays all the information for the configured Global Addressing Validation database for Germany. This example specifies the Global Address Validation International coder.

```
gamdb get info --n GAV_DEU --g Global
```

The returned information may be similar to the following:

```
DATABASE NAME = GAV_DEU
POOL SIZE = 5
BASE_DB_PATH = C:\DBs\DEU\
```

Example

This example returns information in a table for Global Addressing Validation. This example specifies the Global Address Validation International coder.

```
gamdb get info --n GAV --g Global
```

The returned information may be similar to the following:

```
+--------------+---------+---------+
|    SPDNAME   | SPDTYPE | COUNTRY |
+--------------+---------+---------+
|     GAV_APAC |     GAV |     ALL |
|     GAV_EMEA |     GAV |     ALL |
| GAV_AMERICAS |     GAV |     ALL |
+--------------+---------+---------+
```

Example

This example displays all the information for the configured Global Type Ahead database for Austria.

```
gamdb get info --n GTA_AUT
```

The returned information may be similar to the following:

```
DATABASE NAME = GAV_AUT
POOL SIZE = 6
BASE_DB_PATH = C:\DBs\AUT\
```

Example

This example returns information in a table for Global Type Ahead.

```
gamdb get info --n GTA
```

The returned information may be similar to the following:

```
+--------------+---------+---------+
|    SPDNAME   | SPDTYPE | COUNTRY |
+--------------+---------+---------+
|     GTA_APAC |     GTA |     ALL |
|     GTA_EMEA |     GTA |     ALL |
```

```
| GTA_AMERICAS |      GTA |      ALL |
+--------------+---------+----------+
```

gamdb import

The `gamdb import` command imports a Global Addressing database property file that configures the database resources on the current system.

Usage

`gamdb import --f` *File*

Required	Argument	Description
Yes	`--f` *File*	Specifies the JSON-formatted database property file. This file is mandatory.

Example

This example creates a Global Addressing database resource as defined by the configuration provided in the `GlobalAddressingDbResource.txt` JSON-formatted file.

```
gamdb import --f GlobalAddressingDbResource.txt
```

gamdb listdatasets

The `gamdb listdatasets` command displays the Spectrum Global Addressing databases registered on the platform.

Usage

`gamdb listdatasets` This command does not have any properties.

Example

This example lists the Spectrum Global Addressing databases registered on the platform.

```
gamdb listdatasets
```

gamdb listdbresources

The `gamdb listdbresources` command displays all the configured Spectrum Global Addressing databases and the pool size for each database.

Usage

`gamdb listdbresources --g` *Group*

Required	Argument	Description	
Yes	`--g` *Group*	Specifies the coder for Global Address Validation.	
		Global	Global Address Validation International coder.
		US	Global Address Validation US coder.

Example

This example lists the Spectrum Global Addressing databases and the pool size for each database. This example specifies the Global Address Validation International coder.

```
gamdb listdbresources --g Global
```

gamdb memory set

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 272.

The `gamdb memory set` command defines the memory size for Spectrum Global Addressing databases. You must have Spectrum Global Addressing installed to use this command. The fields for defining minimum and maximum memory values can be empty. If a value is empty, that value will not be specified on the command line when starting the component, as if no value were explicitly defined. If no value is specified, or if a value is 0, the property will not be passed to the Command Line Interface.

Usage

`gamdb memory set --name` *database_name* `--mn` *minimum_memory_size* `--mx` *maximum_memory_size*

Note: To see a list of parameters, type `help gamdb memory set`.

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	`--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.

Required	Argument	Description
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 sets the database memory sizes for Spectrum Global Addressing database for EMEA.

```
gamdb memory set --n GAV_EMEA --mn 1200 --mx 65536
```

gamdb modify

The `gamdb modify` command modifies and updates Spectrum Global Addressing databases.

Usage

`gamdb modify --n` *Name* `--d` *Dataset Name* `--v` *Dataset Vintage* `--c` *Country* `--t` *Type* `--g` *Group* `--p` *Poolsize* `--mn` *minimum_memory_size* `--mx` *maximum_memory_size*

Required	Argument	Description
Yes	`--n` *Name*	Specifies the name of the database resource to modify.
Yes	`--d` *Dataset Name*	Specifies the name of the SPD dataset that is the Qualifier name.
Yes	`--v` *Dataset Vintage*	Specifies the vintage of the dataset.
No	`--c` *Country*	Specifies the three-digit ISO code for each country to include in the databases specified by the "t" option (type of SPD) where Countries is a list of three-digit ISO codes separated by semicolons. For more information about ISO codes, see your *Spectrum Technology Platform Addressing Guide*.
Yes	`--t` *Type*	Specifies the type of dataset. **GAV** Global Address Validation database. **GTA** Global Type Ahead database.
Yes	`--g` *Group*	Specifies the coder for Global Address Validation. **Global** Global Address Validation International coder. **US** Global Address Validation US coder.
No	`--p` *Poolsize*	Specifies the maximum number of concurrent requests you want this database to handle. The default is 4.

Required	Argument	Description
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 modifies the poolsize of the Global Addressing Validation database for Germany named "GAV_DEU". This example specifies the Global Address Validation International coder.

```
gamdb modify --n GAV_DEU --d GAV_EMEA --v DEC2018 --c DEU --t
GAV --g Global --p 6 --mn 12200 --mx 65536
```

Example

This example modifies the poolsize of the Global Type Ahead database for Austria named "GTA_AUT".

```
gamdb modify --n GTA_AUT --d GTA_EMEA --v DEC2018 --c AUT --t
GTA --p 3 --mn 12200 --mx 65536
```

gamdb poolsize set

The `gamdb poolsize set` command sets the pool size for a configured Spectrum Global Addressing database resource. The pool size is the maximum number of concurrent requests allowed to a database.

Usage

`gamdb poolsize set --n` *Name* `--s` *Poolsize* `--g` *Group*

Required	Argument	Description
Yes	`--n` *Name*	Specifies the name of the database.
No	`--s` *Poolsize*	Sets the pool size (specified as an integer value) of the database. The default is 4.
Yes	`--g` *Group*	Specifies the coder for Global Address Validation. **Global** Global Address Validation International coder. **US** Global Address Validation US coder.

Example

This example sets the poolsize of an already configured Global Addressing Validation database for Germany to 5. This example specifies the Global Address Validation International coder.

```
gamdb poolsize set --n GAV_DEU --s 5 --g Global
```

Example

This example sets the poolsize of an already configured Global Type Ahead database for Austria to 7.

```
gamdb poolsize set --n GTA_AUT --s 7
```

Spectrum Global Geocoding Databases

globalgeocodedb create sample file

The `globalgeocodedb create_sample_file` command creates sample JSON files of the Spectrum Global Geocoding database resources. These generated files can be used as a reference for providing configurations when creating a database resource. *GeocodeGlobalSingleDictDbResource.txt* and *GlobalGeocodeDoubleDictDbResource.txt* JSON files are created in the current directory or in a specified folder location.

Usage

`globalgeocodedb create_sample_file --o` ***outputpath***

Required	Argument	Description
No	`--o` *outputpath*	The sample database resources JSON files will be created at the designated output directory. If the output path is not specified, the sample JSON files will be written to the current folder.

Example

This example creates the sample database resources JSON files in the current folder.

```
globalgeocodedb create_sample_file
```

Sample JSON database resources file

The following example creates the database resources JSON files in C:\OutputFolder\.

```
globalgeocodedb create_sample_file --o C:\OutputFolder\
```

```
[{"product":"GlobalGeocode",
"module":"GlobalGeocode",
"name":"$$DATABASE_NAME$$",
"maxActive":4,
"properties":
{"COUNTRY_CODE1":"$$COUNTRY_CODE1$$",
 "$$COUNTRY_CODE1$$_DICTIONARY_PATH1":"$$DICTIONARY_PATH1$$",
 "COUNTRY_COUNT":"1",
 "$$COUNTRY_CODE1$$_DICTIONARY_PATH_NAME1":"$$DICTIONARY_PATH_NAME1$$"}
}]
```

globalgeocodedb delete

The `globalgeocodedb delete` command deletes a configured Spectrum Global Geocoding database.

Usage

`globalgeocodedb delete --n` *Name*

Required	Argument	Description
Yes	`--n` *Name*	Specifies the name of the database.

Example

This example deletes the TomTomUSA database.

```
globalgeocodedb delete --n TomTomUSA
```

globalgeocodedb import

The `globalgeocodedb import` command imports a Spectrum Global Geocoding database property file. This configures the database resources on the current system.

Usage

`globalgeocodedb import --f` *File*

Required	Argument	Description
Yes	`--f` *File*	Specifies the JSON-formatted database property file. This file is mandatory.

Example

This example creates a Spectrum Global Geocoding database resource as defined by the configuration provided in the `GlobalGeocodeDbResource.txt` JSON-formatted file.

```
globalgeocodedb import --f GlobalGeocodeDbResource.txt
```

There are several cases that may occur in response to the `globalgeocodedb import` command.

- **Case 1:** The directories in the specified root folder are all invalid. In this case, no database is added.

  ```
  spectrum> globalgeocodedb import --f ./GlobalGeocodeDbResource.txt
  /managers/GlobalGeocode/verify?rootFolder=D:/SGI_Data/
  ```

  The response is as follows:

  ```
  Invalid Folder locations found.
  ["D:\\SGI_Data\\IGEO-AT1"
  "D:\\SGI_Data\\IGEO-CZ1"]
  unable to add the database resource due to invalid paths
  ```

- **Case 2:** The provided root folder has at least one valid directory. In this case, the database is added.

  ```
  spectrum> globalgeocodedb import --f ./GlobalGeocodeDbResource.txt
  /managers/GlobalGeocode/verify?rootFolder=D:/SGI_Data/GEO-DB
  ```

  The response is as follows:

  ```
  Invalid Folder locations found.
  ["D:\\SGI_Data\\IGEO-CZ1"]
  Database resource imported [./GlobalGeocodeDbResource.txt]
  ```

- **Case 3:** The provided root folder is invalid or doesn't exist. In this case, the database is added.

  ```
  spectrum> globalgeocodedb import --f ./GlobalGeocodeDbResource.txt
  ```

  The response is as follows:

  ```
  unable to add the database resource due to invalid paths
  ```

globalgeocodedb export

The `globalgeocodedb export` command exports all of the Spectrum Global Geocoding database resource information to a database properties file, `GlobalGeocodeDbResource.txt`, either at a specified location, or if the location for the output file is not provided, `GlobalGeocodeDbResource.txt` is written to the current folder. The database properties file can

subsequently be used with the `globalgeocodedb import` command to configure the databases on another system.

Usage

`globalgeocodedb export --o` ***outputpath***

Required	Argument	Description
No	`--o` ***outputpath***	The information on Spectrum Global Geocoding database resources will be exported to `GlobalGeocodeDbResource.txt` in the specified output directory. If the path is not provided, `GlobalGeocodeDbResource.txt` will be written to the current folder.

Example

This example exports the Spectrum Global Geocoding database resource information to the designated directory.

```
globalgeocodedb export --o C:\DBs\
```

The `GlobalGeocodeDbResource.txt` output file contains database resource information similar to the following:

```
[{"product":"GlobalGeocode",
"module":"GlobalGeocode",
"name":"TomTomStreets",
"maxActive":4,
"properties":
{"BASE_DB_PATHS":"C:/Dataset/DVDGDT",
"DataSetName":"TomTomStreets"}},
{"product":"GlobalGeocode",
"module":"GlobalGeocode",
"name":"CentrusPoints",
"maxActive":4,
"properties":
{"BASE_DB_PATHS":"C:/Dataset/DVDCPoints;C:/Dataset/DVDGDT",
"DataSetName":"CentrusPoints"}}]
```

globalgeocodedb get

The `globalgeocodedb get` command returns information about a Spectrum Global Geocoding database.

Usage

`globalgeocodedb get --n` ***Name***

Required	Argument	Description
Yes	`--n` *Name*	Specifies the name of the database. This file is mandatory.

Example

This example displays all the information for the configured Spectrum Global Geocoding database resource.

```
globalgeocodedb get --n CENTRUS_PTS
```

The returned information may be similar to the following:

```
DATABASE NAME = CENTRUS_PTS
POOL SIZE = 4
BASE_DB_PATH = C:\DBs\USA\
DataSetName = USA_POINTS
```

globalgeocodedb list

The `globalgeocodedb list` command displays all the configured Spectrum Global Geocoding databases and their pool sizes.

Usage

`globalgeocodedb list` This command does not have any properties.

This example lists the Spectrum Global Geocoding databases and the pool size.

```
globalgeocodedb list
```

globalgeocodedb memory set

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 272.

The `globalgeocodedb memory set` command defines the memory size for Spectrum Global Geocoding databases. You must have Spectrum Global Geocoding installed to use this command. The fields for defining minimum and maximum memory values can be empty. If a value is empty, that value will not be specified on the command line when starting the component, as if no value were explicitly defined. If no value is specified, or if a value is 0, the property will not be passed to the Command Line Interface.

Usage

`globalgeocodedb memory set --name` *database_name* `--mn` *minimum_memory_size* `--mx` *maximum_memory_size*

Note: To see a list of parameters, type `help globalgeocodedb memory set`.

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	`--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 sets the minimum and maximum database memory sizes for the TomTomUSA database.

```
globalgeocodedb memory set --n TomTomUSA --mn 1200 --mx 65536
```

globalgeocodedb poolsize set

The `globalgeocodedb poolsize set` command sets the pool size for a configured Spectrum Global Geocoding database resource. The pool size is the maximum number of concurrent requests allowed to a database.

Usage

`globalgeocodedb poolsize set --n` *Name* `--s` *Poolsize*

Required	Argument	Description
Yes	`--n` *Name*	Specifies the name of the database.
No	`--s` *Poolsize*	Sets the pool size (specified as an integer value) of the database. The default is 4.

Example

This example sets the poolsize of an already configured Spectrum Global Geocoding database resource to 10.

```
globalgeocodedb poolsize set --n DEU_DB -s 10
```

Spectrum Spatial and Routing Databases

limrepo export

> **Note:** For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 272.

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.
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.

Required	Argument	Description
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 272.

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.
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.

Required	Argument	Description
		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. 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

Required	Argument	Description
		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 272.

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.
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

Required	Argument	Description
		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)*
..

ermdb list

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 272.

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 272.

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 272.

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.
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 272.

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 memory set

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 272.

The `ermdb memory set` command defines the memory size for the routing database. You must have Spectrum Spatial installed to use this command. The fields for defining minimum and maximum memory values can be empty. If a value is empty, that value will not be specified on the command line when starting the component, as if no value were explicitly defined. If no value is specified, or if a value is 0, the property will not be passed to the Command Line Interface.

Usage

`ermdb memory set --name` *database_name* `--mn` *minimum_memory_size* `--mx` *maximum_memory_size*

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

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	`--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 sets the memory sizes for a Spectrum Spatial US database.

```
ermdb memory set --name ERM-US --mn 1200 --mx 65536
```

ermdb modify

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 272.

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***.
No	`--vmargs` ***java_argument***	Sets the Java Virtual Machine Arguments (VMArgs) value to add a database to the Spectrum server.

Required	Argument	Description
		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 template

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 272.

The `ermdb template` command creates a JSON file (.json) that is the template for a Spectrum Product Data (SPD) import file. Use this command before using the `ermdb import` command. You must have Spectrum Spatial installed to use these commands.

Usage

`ermdb template` *path_to_JSON*

Required	Argument	Description
No	*path_to_JSON*	Specifies the path to the JSON file to generate a template (.json) file from. By default, the `ermdb template` command generates the JSON file in the current working folder.

Example

This example creates a .json file that acts as a template of information for importing a Spectrum Product Data (SPD) file into a Spectrum database. This example creates the template in the current working folder.

```
ermdb template
```

To export the template to a specific location, include the file name in the command.

```
ermdb templateC:/Downloads/File/A1T_Pedestrian_Mar_2019.json
```

The resulting template file (.json file) has the following content:

```
[{"product":"Spatial",
"module":"routing",
"name":"enter database name",
"properties":{"isSPD":"true",
"DatasetPaths":"${spectrum.spd.Spatial/routing/add IDENTIFIER from productdata
 list};"},
"maxActive":4}
```

To find the `IDENTIFIER` to populate the SPD template with, run the `productdata list` command to list information for the SPD files that you want to import. Locate the `IDENTIFIER` for a specific SPD file to import, and then populate the SPD template (.json file) by adding a database name and the `IDENTIFIER` information.

For example, the following database name is "Austria" (you can type any display name for the data) and the `IDENTIFIER` is "A1T_Pedestrian_Mar_2019":

```
[{"product":"Spatial",
  "module":"routing",
  "name":"Austria",
  "properties":{
     "isSPD":"true",

"DatasetPaths":"${spectrum.spd.Spatial/routing/A1T_Pedestrian_Mar_2019};"},

  "maxActive":4}]
```

You are now ready to run the `ermdb import` command to import these routing data configurations and create the database resources called "Austria" on the Spectrum server.

ermdb import

> **Note:** For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 272.

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 272.

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 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 272.

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 272.

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 272.

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

Required	Argument	Description
		"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.
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 272.

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 272.

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

Required	Argument	Description
		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 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 272.

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 272.

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 272.

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 • major local road rural • major local road suburban

Required	Argument	Description
		• 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: kph (kilometers per hour), mps(meters per second), or mph (miles per hour).

Required	Argument	Description
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 272.

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 484.

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 272.

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 484.

Required	Argument	Description
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 272.

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, 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 272.

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
```

Spectrum Universal Address Databases

uamdb create

The `uamdb create` command creates a new Spectrum Universal Addressing database.

Usage

`uamdb create --t` *Type* `--n` *Name* `--c` *CacheSize* `--i` *Country* `--pl` *PreloadingType* `--dt` *DatabaseType* `--b` *BasePath* `--d` *DPVPath* `--l` *LACSPath* `--s` *SuiteLinkPath* `--r` *RDIPath* `--e` *EWSPath* `--p` *Poolsize* `--mm` *minimumMemorySize* `--mx` *maximumMemorySize*

Note: To see a list of parameters, type `help uamdb create`.

Required	Argument	Description
Yes	`--t` *Type*	Specifies the type of database, where *Type* is one of the following: **USA** United States database **CAN** Canadian Database **INTL** International Database **Loqate** Loqate Database
Yes	`--n` *Name*	Specifies the name of the database.
Yes	`--b` *BasePath*	Specifies the base subscription database path. **Note:** For **USA**, **CAN**, **INTL**, **Loqate** specify the database vintage in place of database path. **Example:** `NOV2017`
No	`--d` *DPVPath*	Specifies the DPV database vintage.
No	`--l` *LACSPath*	Specifies the LACS database vintage.
No	`--s` *SuiteLinkPath*	Specifies the SuiteLink database vintage.
No	`--r` *RDIPath*	Specifies the RDI database vintage.
No	`--e` *EWSPath*	Specifies the EWS database vintage.
No	`--p` *Poolsize*	Specifies the maximum number of concurrent requests you want this database to handle. The default is 4.
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.

Note: The *database vintage* can be obtained using the *uamdb listdatasets* command. For more information see **uamdb listdatasets** on page 496.

Example

To create a database for *UAM US*, *CAN*, *INTL*, *Loqate* provide input in this format:

```
uamdb create --t USA --n UAM_US --b FEB2018 --d AUG2018 --r
SEP2018 --mn 1200 --mx 65536
```

uamdb modify

The `uamdb modify` command updates an existing Spectrum Universal Addressing database.

Usage

`uamdb modify --t` *Type* `--n` *Name* `--b` *BasePath* `--d` *DPVPath* `--l` *LACSPath* `--s` *SuiteLinkPath* `--r` *RDIPath* `--e` *EWSPath* `--p` *Poolsize* `--mm` *minimumMemorySize* `--mx` *maximumMemorySize*

Required	Argument	Description
Yes	`--t` *Type*	Specifies the type of database, where *Type* is one of the following: **USA** United States database **CAN** Canadian Database **INTL** International Database **Loqate** Loqate Database
Yes	`--n` *Name*	Specifies the name of the database.
Yes	`--b` *BasePath*	Specifies the base subscription database path. **Note:** For **USA**, **CAN**, **INTL**, **Loqate** specify the database vintage in place of database path. **Example:** `NOV2017`
No	`--d` *DPVPath*	Specifies the DPV database vintage.
No	`--l` *LACSPath*	Specifies the LACS database vintage.
No	`--s` *SuiteLinkPath*	Specifies the SuiteLink database vintage.
No	`--r` *RDIPath*	Specifies the RDI database vintage.
No	`--e` *EWSPath*	Specifies the EWS database vintage.
No	`--p` *Poolsize*	Specifies the maximum number of concurrent requests you want this database to handle. The default is 4.
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.

Note: The *database vintage* can be obtained using the *uamdb listdatasets* command. For more information see **uamdb listdatasets** on page 496.

Example

To modify a database for *UAM US*, *CAN*, *INTL*, *Loqate* provide input in this format:

```
uamdb modify --n UAM_US --t USA --b SEP2018 --d AUG2018 --r
OCT2018 --mn 1200 --mx 65536
```

uamdb delete

The `uamdb delete` command deletes a Spectrum Universal Addressing database.

Usage

`uamdb delete --t` *Type* `--n` *Name*

Required	Argument	Description	
Yes	`--t` *Type*	Specifies the type of database, where *Type* is one of the following:	
		USA	United States database
		CAN	Canadian Database
		INTL	International Database
		Loqate	Loqate Database
		Amas	Australian Database
Yes	`--n` *Name*	Specifies the name of the database.	

Example

This example deletes a Canadian database named "UAM_CAN".

```
uamdb delete --t CAN --n UAM_CAN
```

uamdb memory set

Note: For instructions on installing and running the Administration Utility, see **Getting Started with the Administration Utility** on page 272.

The `uamdb memory set` command defines the memory size for the Spectrum Universal Addressing database. You must have Spectrum Universal Addressing installed to use this command. The fields for defining minimum and maximum memory values can be empty. If a value is empty, that value will not be specified on the command line when starting the component, as if no value were explicitly defined. If no value is specified, or if a value is 0, the property will not be passed to the Command Line Interface.

Usage

`uamdb memory set --name` ***database_name*** `--mn` ***minimum_memory_size*** `--mx` ***maximum_memory_size***

Note: To see a list of parameters, type `help uamdb memory set`.

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	`--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 sets the database memory sizes for a Spectrum Universal Addressing U.S. database.

```
uamdb memory set --name UAM_US --mn 1200 --mx 65536
```

uamdb import

The `uamdb import` command imports a Spectrum Universal Addressing database.

Usage

`uamdb import --t` ***Type***

Required	Argument	Description	
Yes	`--t` *Type*	Specifies the type of database, where *Type* is one of the following:	
		USA	United States database
		CAN	Canadian Database
		INTL	International Database
		Loqate	Loqate Database
		Amas	Australian Database

Example

This example imports a United States database.

```
uamdb import --t USA
```

uamdb export

The `uamdb export` command exports a Spectrum Universal Addressing database.

Usage

`uamdb export --t` *Type*

Required	Argument	Description	
Yes	`--t` *Type*	Specifies the type of database, where *Type* is one of the following:	
		USA	United States database
		CAN	Canadian Database
		INTL	International Database
		Loqate	Loqate Database
		Amas	Australian Database

Example

This example exports an international database.

```
uamdb export --t INTL
```

uamdb get resource info

The `uamdb get resource info` command returns information about a database.

Usage

`uamdb get resource info --t` *Type* `--n` *Name*

Required	Argument	Description	
Yes	`--t` *Type*	Specifies the type of database, where *Type* is one of the following:	
		USA	United States database
		CAN	Canadian Database
		INTL	International Database
		Loqate	Loqate Database

Required	Argument	Description	
		Amas	Australian Database
Yes	`--n` *Name*	Specifies the name of the database.	

Example

This example retrieves information for a United States database named "UAM_US".

```
uamdb get resource info --t USA --n UAM_US
```

It may return information similar to the following:

```
DATABASE NAME = UAM_US
POOL SIZE = 4
LACS_DB_PATH = Z:\UAM\US_AUG12
SUITELINK_DB_PATH = Z:\UAM\US_AUG12
BASE_DB_PATH = Z:\UAM\US_AUG12
DPV_DB_PATH = E:\UAM_US_MAY_14_DB
RDI_DB_PATH = E:\UAM_US_MAY_14_DB
EWS_DB_PATH = Z:\UAM\US_AUG12
```

uamdb list

The `uamdb list` command returns all Spectrum Universal Addressing databases of that type in tabular format.

Usage

`uamdb list --t` *Type*

Required	Argument	Description	
Yes	`--t` *Type*	Specifies the type of database, where *Type* is one of the following:	
		USA	United States database
		CAN	Canadian Database
		INTL	International Database
		Loqate	Loqate Database

Example

This example lists all Canadian databases.

```
uamdb list --t CAN
```

uamdb listdatasets

The `uamdb listdatasets` command displays the Spectrum Universal Addressing databases registered on the platform. Details such as **Component**, **Name**, and **Vintage** are displayed here.

Usage

`uamdb listdatasets --t` *Type*

Required	Argument	Description	
Yes	`--t` *Type*	Specifies the type of database, where *Type* is one of the following:	
		USA	United States database
		CAN	Canadian Database
		INTL	International Database

This example lists the United States databases registered on the platform.

```
uamdb listdatasets --t USA
```

uamdbglobalmultipath create_sample_file

The `uamdbglobalmultipath create_sample_file` command configures a database resource with multiple path elements and creates a sample JSON file (`UamDbGlobalMultiPath.txt`) that can be modified with place holders and data paths. This command should be followed by a `uamdb_import` command for additional database configuration.

Note: You must replace the token values in the text file with the absolute values and data paths.

Usage

`uamdbglobalmultipath create_sample_file --o` *OutputDirectory* `--n` *NumberOfPathElements*

Required	Argument	Description
No	`--o` *OutputDirectory*	Specifies the directory where the file should go. The current directory is the default location.
Yes	`--n` *NumberOfPathElements*	Specifies the number of elements in the path.

Example

This example creates a sample *JSON* file named *"UamDbGlobalMultiPath.txt"* with its properties in *JSON* key-value format. This database resource has three path elements.

```
uamdbglobalmultipath create_sample_file --n 3
```

uamdb poolsize set

The `uamdb poolsize set` command sets the default poolsize for a database.

Usage

`uamdb poolsize set --t` *Type* `--n` *Name* `--s` *Size*

Required	Argument	Description
Yes	`--t` *Type*	Specifies the type of database, where *Type* is one of the following: **USA** United States database **CAN** Canadian Database **INTL** International Database
Yes	`--n` *Name*	Specifies the name of the database.
Yes	`--s` *Size*	Specifies the default poolsize.

Example

This example sets the poolsize of a Canadian database named "UAM_CAN" to 4.

```
uamdb poolsize set --t CAN --n UAM_CAN --s 4
```

Service pool size

service poolsize default get

The `service poolsize default get` command returns detailed pool size default information for all services. The pool size is an integer value representing the maximum number of concurrent requests allowed to a resource pool.

Usage

```
service poolsize default get
```

service poolsize default set

The `service poolsize default set` command defines the default pool size for all configured dataflow services. The pool size is an integer value representing the maximum number of concurrent requests allowed to a resource pool.

Usage

`service poolsize default set --p` ***poolSize***

Required	Argument	Description
Yes	`--p` ***poolSize***	Sets the default pool size (specified as an integer value) for the service. There is no default. Work with your system administrator or a Precisely support representative if you need help in setting this value.

Example

This example sets the poolsize of a configured Global Addressing Validation service for Germany (GAV_DEU) to 5.

```
service poolsize default set --p 5
```

service poolsize get

The `service poolsize get` command returns detailed pool size information about a dataflow service resource. The pool size is an integer value representing the maximum number of concurrent requests allowed to a resource pool.

Usage

`service poolsize get --s` ***serviceName***

Required	Argument	Description
Yes	`--s` ***serviceName***	Specifies the service details to list.

Example

This example returns information about a user-defined parsing address service, "ParsingAddresses."

```
service poolsize get --s ParsingAddresses
```

service poolsize set

The `service poolsize set` command defines a pool size limit for a dataflow service resource. The pool size is an integer value representing the maximum number of concurrent requests allowed to a resource pool.

Usage

`service poolsize set --s` *serviceName* **--p** *poolSize*

Required	Argument	Description
Yes	`--s` *serviceName*	Specifies the service name to which this definition applies.
Yes	**--p** *poolSize*	Sets the pool size (specified as an integer value) for the service. There is no default. Work with your system administrator or a Precisely support representative if you need help in setting this value.

Example

This example defines the default pool size for a user-defined parsing address service, "ParsingAddresses."

```
service poolsize set --s ParsingAddresses --p 4
```

System

close

The `close` command closes the session with the Spectrum Technology Platform server. Use this command if you want to close the session to the server without exiting the Administration Utility. You can close and exit by using the `exit` command.

Usage

```
close
```

connect

The `connect` command opens a session with the Spectrum Technology Platform server you specify. You must issue the `connect` command before you can issue other commands.

Usage

`connect --h` *HostName* `--u` *UserName* `--p` *Password* `--s` *TrueOrFalse*

Required	Argument	Description
Yes	`--h` *HostName*	The host name and port to connect to, separated by a colon. For example, to connect to MyServer on port 8080 you would specify `--h MyServer:8080`.
Yes	`--u` *UserName*	The user name to use to authenticate to the server.
Yes	`--p` *Password*	The password for the user.
No	`--s` *TrueOrFalse*	Specifies whether to create a secure connection using HTTPS. You can only connect to the server over a secure connection if the server has been configured to support HTTPS communication. *TrueOrFalse* must be one of the following: **true** Use HTTPS. **false** Do not use HTTPS. This is the default setting.

Example

This example opens a connection to the server MyServer on port 8080 with the user name admin and the password myPassword1.

```
connect --h MyServer:8080 --u admin --p myPassword1
```

date

The `date` command displays the current date and time on the computer where you are running the Administration Utility.

Usage

```
date
```

exit

The `exit` command closes the session and exits the Administration Utility. If you want to close your session without exiting the Administration Utility, use the `close` command.

Usage

```
exit
```

help

The `help` command displays a list of the commands you can use in the Administration Utility. You can also use the `help` command to get information about the parameters used in each command.

Usage

`help` *Command*

Required	Argument	Description
No	*Command*	If you specify a command name as a parameter to the `help` command, detailed information about the command you specify is displayed. If you do not specify a command name, a list of all commands is shown.

Example

This command lists all the commands available in the Administration Utility:

```
help
```

This command displays detailed information about the `set serviceoption` command:

```
help set serviceoption
```

license expirationinfo export

The `license expirationinfo export` command exports a list of licenses that are about to expire. The licenses that are included are those that are going to expire within the time specified in the Spectrum Management Console on the **Notification** tab.

Usage

`license expirationinfo export --o` *Directory*

Required	Argument	Description
No	`--o` *Directory*	Specifies the directory to which you want to export the license expiration information. The path you specify here is relative to the directory where you are running the Administration Utility. If you omit this option the expiration information is placed in a file in the folder where you are running the Administration Tool.

Example

This example exports license expiration information to the LicenseExpiration subfolder under the folder where the Administration Tool is located.

```
license expirationinfo export --o LicenseExpiration
```

license expirationinfo list

The `license expirationinfo list` command returns a list of licenses that are about to expire. The licenses that are displayed are those that are going to expire within the time specified in the Spectrum Management Console on the **Notification** tab.

Usage

```
license expirationinfo list
```

licenseinfo export

The `licenseinfo export` command exports license information to a file. A license file may be needed when resolving license issues with technical support.

Usage

```
licenseinfo export --o Directory
```

Required	Argument	Description
No	`--o` *Directory*	Specifies the directory to which you want to export the license file. The path you specify here is relative to the directory where you are running the Administration Utility. If you omit this option the license file is placed in the folder where you are running the Administration Tool.

Example

This example exports license information to the License subfolder under the folder where the Administration Tool is located.

```
licenseinfo export --o License
```

licenseinfo list

The `licenseinfo list` command displays license information such as which licenses are installed, the number of days remaining on the license, and the number of transactions remaining.

Usage

```
licenseinfo list
```

server backup

Use the `server backup` command to back up your Spectrum Technology Platform server.

To back up your Spectrum Technology Platform server, you need to create a backup copy of the server's configuration database. The configuration database contains your security settings, dataflows, service options, data resource definitions, snapshots, and various configuration settings. If you were to lose your server due to a system failure or other disaster, you could use the backup of the configuration database to restore your configuration to another Spectrum Technology Platform server.

Important: Do not run a backup when there is activity on the Spectrum Technology Platform server. While the backup is in progress, calls to services may time out and jobs may fail to execute successfully.

Usage

```
server backup --o Directory
```

Required	Argument	Description
No	`--o` *Directory*	Specifies the directory to which you want to save the backup copy of the server's database. The path you specify here is relative to the directory where you are running the Administration Utility. If you omit this option the backup is placed in the folder where you are running the Administration Tool.

Example

This example saves the backup to the LatestServerBackup subfolder under the folder where the Administration Tool is located.

```
server backup --o LatestServerBackup
```

Notes

The backup process creates a temporary (`.temp`) directory, and should a backup fail, it creates an error (`.err.*`) directory. After you run the `server backup` command, it is safe to remove these directories to free up drive space. Removing these directories will not affect the restore process.

Related tasks

Restore a Spectrum repository backup on page 256
Use this procedure to restore the Spectrum respository from a backup created by the Admiinistration Uitility `server backup` CLI command.

script

The `script` command directs the Administration Tool to execute a script containing a series of commands. You can use a script to automate administrative tasks.

Usage

`script --file` *ScriptFile* `--linenumbers` *TrueOrFalse*

Required	Argument	Description
Yes	`--file` *ScriptFile*	Specifies the path to the script file.
No	`--linenumbers` *TrueOrFalse*	Specifies whether to display line numbers while executing the script, where *TrueOrFalse* is one of the following: **true** Displays line numbers while executing the script. **false**

Required	Argument	Description
		Does not display line numbers while executing the script. This is the default value.

Example

This example executes a script named `myscript.cli` located in the folder `scripts` which is a subfolder under the folder that contains the Administration Utility.

```
script --file scripts/myscript.cli
```

system loglevel get

The `system loglevel get` command returns the default logging level for services. The logging levels are:

Off	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.

Usage

```
system loglevel get
```

system loglevel set

The `system loglevel set` command sets the default logging level for services on your system.

Usage

`system loglevel set --l` *Level*

Required	Argument	Description
Yes	`--l` *Level*	Specifies the default logging level for services on your system, where *Level* is one of the following: **Off** — 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. **Note:** Selecting the most intensive logging level can affect system performance. Therefore, you should select the least intensive setting that meets your particular logging requirements.

Example

This example sets the default logging level to Warn:

```
system loglevel set --l warn
```

system properties

The `system properties` command displays information about the shell running the Administration Utility, such as Java properties and OS version. It does not display information about the Spectrum Technology Platform server.

Usage

```
system properties
```

versioninfo export

The `versioninfo export` command exports system, component, and service version information to a file.

Usage

`versioninfo export --o` *Directory*

Required	Argument	Description
No	`--o` *Directory*	Specifies the directory to which you want to export the version information text file. The path you specify here is relative to the directory where you are running the Administration Utility. If you omit this option the version information file is placed in the folder where you are running the Administration Tool.

Example

This example exports version information to the VersionInformation subfolder under the folder where the Administration Tool is located.

```
versioninfo export --o VersionInformation
```

versioninfo list

The `versioninfo list` command displays information about the version of Spectrum Technology Platform installed on your system, its underlying components, as well some system information.

Usage

```
versioninfo list
```

Tables

table delete

The `table delete` command removes a table from your system. For more information, see the "Lookup Tables" section of the *Data Quality Guide*.

Usage

`table delete` *TableName* `--t` *TableType*

Required	Argument	Description
Yes	`--n` *TableName*	Specifies the table to delete.
Yes	`--t` *TableType*	Specifies the type of table to delete: AdvancedTransformer, OpenParser, or TableLookup.

Example

This example deletes the Table Lookup table named My Table.

```
table delete My Table --t TableLookup
```

table export

The `table export` command exports a custom table that was created using the Table Management feature in Spectrum Enterprise Designer. The table can then be imported to another server. For more information, see the "Lookup Tables" section of the *Data Quality Guide*.

Usage

`table export` *TableName* `--t` *TableType* `--o` *OutputDirectory* `--f` *ExportedFileName*

Required	Argument	Description
Yes	`--n` *TableName*	Specifies the name of the table you want to export. **Tip:** If you are unsure of the exact table name you can use the `table list` command to get a list of the table names.
Yes	`--t` *TableType*	Specifies the type of table to export: AdvancedTransformer, OpenParser, or TableLookup.
No	`--o` *OutputDirectory*	Specifies the directory to which you want to export the table. The path you specify here is relative to the directory where you are running the Administration Utility. If you omit this argument, the table is exported to the directory containing the Administration Utility.
No	`--f` *ExportedFileName*	Specifies the name of the file to be exported. If the table name you wish to export contains ' / ' character, you can rename it using this option and further export it.

Example

This example exports an Open Parser table named "My Table" to the location where you have installed the Administration Utility.

```
table export --n My Table --t OpenParser
```

Example

This example exports an Open Parser table named *"AC/E"* by renaming it to *"AC_E"* using the *--f* option to the location where you have installed the Administration Utility.

```
table export --n AC/E --t OpenParser --f AC_E
```

table import

The `table import` command imports a custom table into the server. Custom tables are created using the Table Management feature in Spectrum Enterprise Designer. For more information, see the "Lookup Tables" section of the *Data Quality Guide*.

Usage

`table import` *CustomTable* `--u` *TrueOrFalse*

Required	Argument	Description
Yes	`--f` *CustomTable*	Specifies the custom table you want to import. Directory paths you specify here are relative to the location where you are running the Administration Utility.

Required	Argument	Description
No	`--u` *TrueOrFalse*	Specifies whether to overwrite the existing table if a table with the same name is already on the server, where *TrueOrFalse* is one of the following: **true** If there is a table on the server with the same name as the table you are importing, the table on the server will be overwritten. This is the default setting. **false** If there is a table on the server with the same name as the table you are importing, the table will not be imported.

Example

This example imports the table named `MyTable.db` which is located in a subfolder named `exported` in the location where you are running the Administration Utility. Because no --u command is specified, any existing table of the same name on the server will be overwritten.

```
table import exported\MyTable.db
```

table list

The `table list` command lists all the tables on the server. For each table, the following information is displayed: the table name and whether the dataflow is exposed.

Usage

`table list --t` *TableType*

Required	Argument	Description
Yes	`--t` *TableType*	Specifies the type of tables to list: AdvancedTransformer, OpenParser, or TableLookup.

Example

This example lists all Advanced Transformer tables.

```
table list --t AdvancedTransformer
```

Tokens

token list

The `token list` command returns a list of the active tokens on the Spectrum Technology Platform server. For each token, the following information is provided:

- User name
- Date and time the token was created
- Date and time when the token was last used
- The IP address of the computer being used to access the Spectrum Technology Platform server
- The session ID
- The token

Each of these fields is separated by a pipe character (|).

Usage

`token list --u` *UserName*

Required	Argument	Description
No	`--u` *UserName*	Specifies the user whose tokens you want to view. If you do not specify this argument, all users' tokens are listed.

Example

This example lists the tokens for the user amy123.

```
token list --u amy123
```

This example lists all tokens.

```
token list
```

token refreshsecret

The `token refreshsecret` command refreshes the secret key. This has the effect of rendering all active tokens invalid. Users with active tokens will need to log in again to obtain a new token.

Usage

```
token refreshsecret
```

token revoke

The `token revoke` command make a token invalid. The user will need to log in again to obtain a new token.

Usage

```
token revoke --t Token
```

Required	Argument	Description
Yes	`--t` *Token*	Specifies the token you want to revoke. To see a list of active tokens, use the `token list` command.

Example

This example revokes the token specified in the `--t` argument.

```
token revoke --t
```

token userrevoke

The `token userrevoke` command makes all of a user's tokens invalid. The user will need to log in again to obtain a new token.

Usage

```
token userrevoke --u UserName
```

Required	Argument	Description
Yes	`--u` *UserName*	Specifies the user whose tokens you want to revoke. To see a list of active users, use the `token list` command.

Example

This example revokes the tokens for the user amy123.

```
token userrevoke --u amy123
```

User Accounts

user create

The `user create` command creates a new user and assigns roles to it.

Usage

`user create --u` *UserName* `--p` *Password* `--d` *Description* `--e` *EmailAddress* `--r` *Roles*

Required	Argument	Description
Yes	`--u` *UserName*	Specifies the name of the new user.
Yes	`--p` *Password*	Specifies the password for the new user.
No	`--d` *Description*	Specifies a description for the user. If the description contains spaces, enclose the description in quotes.
No	`--e` *EmailAddress*	Specifies the email address of the new user.
No	`--r` *Roles*	Specifies the roles for the user. Separate multiple roles with a comma. Do not use spaces. If the role name contains a space, enclose the role in quotes.

Example

This example creates a new user named allan12, assigns a password of myPassword1, a description of Allan P. Smith, an email of `allan@example.com`, and assigns two roles, USBanking and California Users.

```
user create --u allan12 --p myPassword1 --d "Allan P. Smith"
--e allan@example.com --r USBanking,"California Users"
```

user delete

The `user delete` command removes a user account from the system.

Tip: User accounts can also be disabled, which prevents the system access without deleting the account.

Usage

`user delete --u` *UserName*

Required	Argument	Description
Yes	`--u` *UserName*	Specifies the name of the user you want to delete.

Example

This example deletes a user named allan12.

```
user delete --u allan12
```

user description set

The `user description set` command changes the account description.

Usage

`user description set --u` *UserName* `--d` *Description*

Required	Argument	Description
Yes	`--u` *UserName*	Specifies the user account whose description you want to change.
Yes	`--d` *Description*	Specifies a description for the user. If the description contains spaces, enclose the description in quotes.

Example

This example changes the description of the user allan12 to "Allan P. Smith."

```
user description set --u allan12 --d "Allan P. Smith"
```

user email set

The `user email set` command changes the email address associated with the user.

Usage

`user email set --u` *UserName* `--e` *EmailAddress*

Required	Argument	Description
Yes	`--u` *UserName*	Specifies the user account whose email address you want to change.
Yes	`--e` *EmailAddress*	Specifies the email address to associate with the user.

Example

This example sets the email address for the user account allan12 to `allan@example.com`.

```
user email set --u allan12 --e allan@example.com
```

user enabled set

The `user enabled set` command enables or disables a user account.

Note: You cannot disable the user account "admin".

Usage

`user enabled set --u` *UserName* `--e` *TrueOrFalse*

Required	Argument	Description
Yes	`--u` *UserName*	Specifies the name of the user you want to disable.
Yes	`--e` *TrueOrFalse*	Specifies whether to enable or disable the user account where *TrueOfFalse* is one of the following: **true** Enables the user account. **false** Disables the user account.

Example

This example disables the user account allan12.

```
user enabled set --u allan12 --e false
```

user list

The `user list` command returns a list of users. For each user, the command lists the user's roles, email address, description, and whether the user is enabled or disabled.

Usage

```
user list
```

user password set

The `user password set` command changes the password for a user account.

Usage

`user password set --u` *UserName* `--p` *Password*

Required	Argument	Description
Yes	`--u` *UserName*	Specifies the user account whose password you want to change.
Yes	`--p` *Password*	Specifies the password to assign to the user account.

Example

This example sets the password for the user account allan12 to mypassword1.

```
user password set --u allan12 --p mypassword1
```

user role grant

The `user role grant` command assigns one or more roles to a user.

Usage

`user role grant --u` *UserName* `--r` *Roles*

Required	Argument	Description
Yes	`--u` *UserName*	Specifies the name of the user to whom you want to assign a role.
Yes	`--r` *Roles*	Specifies the roles for the user. Separate multiple roles with a comma. Do not use spaces. If the role name contains a space, enclose the role in quotes.

Example

This example assigns two roles, USBanking and CaliforniaUsers, to the user allan12.

```
user role grant --u allan12 --r USBanking,CaliforniaUsers
```

user role list

The `user role list` command returns a list of roles on your system.

Usage

```
user role list
```

user role revoke

The `user role revoke` command removes a role from a user so that the user no longer has the privileges granted by the role.

Usage

`user role revoke --u` *UserName* `--r` *Roles*

Required	Argument	Description
Yes	`--u` *UserName*	Specifies the name of the user whose role you want to revoke.
Yes	`--r` *Roles*	Specifies the role you want to revoke. Separate multiple roles with a comma. Do not use spaces. If the role name contains a space, enclose the role in quotes.

Example

This example revokes the role USBanking from the user allan12.

```
user role revoke --u allan12 --r USBanking
```

13 - Spectrum properties

In this section

spectrum-container.properties reference.....520
Configure proxy server settings.....539

spectrum-container.properties reference

This section provides a reference to properties in the `spectrum-container.properties` file, located in *`SpectrumDirectory`*`/server/conf/spectrum-container.properties`.

> **Note:** Properties prefaced with the `#` symbol in the properties file are commented out in the properties file. We recommend that you work with Precisely Technical Support before changing any properties that are commented out.

- **Server settings** on page 520
- **Cluster settings** on page 521
- **SSL settings** on page 521
- **Data Federation Model Store SSL settings** on page 523
- **Cache settings** on page 524
- **HTTP settings** on page 525
- **CORS** on page 527
- **Other HTTP** on page 528
- **Configuration settings** on page 529
- **Transaction settings** on page 529
- **Runtime settings** on page 529
- **Repository settings** on page 530
- **Repository backup settings** on page 530
- **Index settings** on page 532
- **Debug manager settings** on page 533
- **Security settings** on page 534
- **Data manager settings** on page 535
- **Online help site** on page 535
- **Audit archive settings** on page 535
- **Advanced settings** on page 535
- **Metadata settings** on page 537
- **Other settings** on page 539

Server settings

Property	Default value	More information
`spectrum.agent.address`	None. Use your site's agent address	

Property	Default value	More information
`spectrum.bind.address`	None. Use your site's default binding address/URL	

Cluster settings

Property	Default value	More information
`spectrum.cluster.enabled`	`true`	Cluster Properties
`spectrum.cluster.name`	`SpectrumCluster`	
`spectrum.cluster.password`	*`encrypted string`*	
`spectrum.cluster.seeds`	`127.0.0.1`	Upgrading a cluster, Cluster Properties
`spectrum.cluster.port`	`5701`	Each node in the cluster must have a unique integer `nodeId`
`spectrum.cluster.nodeId`	`1`	Each node in the cluster must have a unique integer `nodeId`.
`spectrum.socketgateway.port`	`10119`	Network Ports

SSL settings

Property	Default value	More information
`spectrum.encryption.enabled`	`false`	Encryption Methods. **Note:** By default, encryption is disabled for all portions of the server, including HTTP/HTTPS, indexing, model store, and caching. Set this property as *"true"* to enable encryption for all portions of the server using the global settings.

Property	Default value	More information
spectrum.encryption.algorithm	JASYPT	
spectrum.encryption.keystoreAlias	spectrum	Encryption Methods
spectrum.encryption.keystoreType	pkcs12	Encryption Methods
spectrum.encryption.keystore	../conf/certs/spectrum-keystore.p12	Encryption Methods
spectrum.encryption.keystorePassword	*encrypted string*	Encryption Methods
spectrum.encryption.truststoreType	pkcs12	Encryption Methods
spectrum.encryption.truststore	../conf/certs/spectrum-truststore.p12	Encryption Methods
spectrum.encryption.truststorePassword	*encrypted string*	Encryption Methods
spectrum.encryption.validateCerts	true	Set to true to implement self-signed certificates in Spectrum Technology Platform.
spectrum.encryption.trustAllHosts	false	Set to true to implicitly trust all certificates; during verification, ignore host name specified on certificate
spectrum.encryption.selfSignedCert	false	Set to true to bypass CA trust validation.

Data Federation Model Store SSL settings

Property	Default value	More information
`spectrum.modelstore.encryption.type.twoway`	`false`	Encryption Methods. **Note:** Set it to *true* to enable two-way SSL mode. If set to *false*, one way SSL mode is enabled.
`spectrum.modelstore.encryption.truststoreCheckExpired`	`true`	Set to true to check the truststore certificate expiry.
`spectrum.modelstore.encryption.algorithm`	`JASYPT`	
`spectrum.modelstore.encryption.keystoreAlias`	`spectrum`	Encryption Methods
`spectrum.modelstore.encryption.keystoreType`	`pkcs12`	Encryption Methods
`spectrum.modelstore.encryption.keystore`	`../conf/certs/spectrum-keystore.p12`	Encryption Methods
`spectrum.modelstore.encryption.keystorePassword`	*`encrypted string`*	

Property	Default value	More information
		Encryption Methods For more information: • Review **Generate encryption strings** on page 88 for information about generating encryption strings. • Review **Encrypt passwords or mask encryption strings** on page 89 for more information about encrypting or masking strings.
`spectrum.modelstore.encryption.truststore`	`../conf/certs/spectrum-truststore.p12`	Encryption Methods
`spectrum.encryption.truststorePassword`	*`encrypted string`*	Encryption Methods For more information: • Review **Generate encryption strings** on page 88 for information about generating encryption strings. • Review **Encrypt passwords or mask encryption strings** on page 89 for information about encrypting or masking strings.

Cache settings

Property	Default value	More information
`spectrum.cache.encryption.enabled`	`false`	Encryption Methods - Caching properties
`spectrum.cache.encryption.keystoreType`	`pkcs12`	Encryption Methods - Caching properties

Property	Default value	More information
spectrum.cache.encryption.keystore	../conf/certs/spectrum-keystore.p12	Encryption Methods - Caching properties
spectrum.cache.encryption.keystorePassword	*encrypted string*	Encryption Methods - Caching properties
spectrum.cache.encryption.truststoreType	pkcs12	Encryption Methods - Caching properties
spectrum.cache.encryption.truststore	../conf/certs/spectrum-truststore.p12	Encryption Methods - Caching properties
spectrum.cache.cache.encryption.truststorePassword	*encrypted string*	Encryption Methods - Caching properties
spectrum.cache.encryption.trustAllHosts	false	
spectrum.cache.encryption.selfSignedCerts	false	
spectrum.cache.mancenter.url	http://127.0.0.1:8090/mancenter	
spectrum.cache.slow.operation.detection.enabled	false	
spectrum.cache.slow.operation.detection.log.enabled	false	
spectrum.cache.slow.operation.detection.threshold	300	

HTTP settings

Property	Default value	More information
spectrum.http.default.protocol	http	
spectrum.http.enabled	true	
spectrum.http.port	8080	Network port for HTTP
spectrum.https.enabled	false	Set to true to enable HTTPS.
spectrum.https.port	8443	Network port for HTTPS.
spectrum.https.encryption.keystoreType	pkcs12	
spectrum.https.encryption.keystore	../conf/certs/spectrum-keystore.p12	
spectrum.https.encryption.keystorePassword	*encrypted string*	
spectrum.https.encryption.keystoreAlias	spectrum	
spectrum.https.encryption.truststoreType	pkcs12	

Property	Default value	More information
spectrum.https.encryption.truststore	../conf/certs/spectrum-truststore.p12	
spectrum.https.encryption.truststorePassword	*encrypted string*	
spectrum.https.encryption.validateCerts	true	Set this to `false` for a self-signed certificate. **Note:** This property is set to `true` after an upgrade to the 20.1 version of Spectrum Technology Platform even if it was previously set to `false`.
spectrum.https.encryption.selfSignedCert	false	Set to `true` to use a self-signed certificate.
spectrum.https.encryption.trustAllHosts	false	

CORS

Property	Default value	More information
spectrum.http.cors.enabled	false	

Property	Default value	More information
spectrum.http.cors.allowedOrigins	http://127.0.0.1:8080,http://127.0.0.1:443	
spectrum.http.cors.allowedMethods	POST,GET,OPTIONS,PUT,DELETE,HEAD	
spectrum.http.cors.allowedHeaders	X-PINGOTHER, Origin, X-Requested-With, Content-Type, Accept	
spectrum.http.cors.preflightMaxAge	1800	
spectrum.http.cors.allowCredentials	false	

Other HTTP

Property	Default value	More information
spectrum.http.sendVersion	false	
spectrum.http.log.request.enabled	false	
spectrum.http.cache.control.headers.enable	true	
spectrum.http.useFileMappedBuffer	false	
spectrum.http.allowDirectoryListing	false	
spectrum.http.minimumThreads	10	
spectrum.http.maximumThreads	250	
spectrum.http.client.maxConnectionsPerHost	32	
spectrum.http.client.maxTotalConnections	128	
spectrum.http.request.header.size	8192	

Property	Default value	More information
spectrum.http.response.header.size	8192	

Configuration settings

Property	Default value	More information
spectrum.configuration.hostname		
spectrum.configuration.port		

Transaction settings

Property	Default value	More information
spectrum.transaction.hostname		
spectrum.transaction.port		

Runtime settings

Property	Default value	More information
spectrum.runtime.hostname		The fully qualified domain name for the host machine.
spectrum.runtime.port		Specifies the port (for example, 8443 when configuring Spectrum for SSL).

Repository settings

Property	Default value	More information
`spectrum.repository.addresses`	`127.0.0.1`	Comma separated list of *`host`*`:`*`port`* pairs indicating members of cluster. If port not specified it defaults to default repository port.
`spectrum.repository.port`	`7687`	
`spectrum.repository.username`	`neo4j`	
`spectrum.repository.password`	*`encrypted string`*	
`spectrum.repository.pool.size`	`50`	
`spectrum.repository.timeout`	`1200`	
`spectrum.repository.cluster.mode`	`CASUAL`	The default setting is for a cluster configuration. Set this to `HA` for a stand-alone server or database configuration.
`spectrum.repository.server.seeds`	`127.0.0.1`	
`spectrum.repository.server.path`	`${g1.server.dir}/../repository`	
`spectrum.repository.server.startup.timeout`	`120`	

Repository backup settings

Property	Default value	More information
`spectrum.backup.enabled`	`false`	Enable or disable backups.

Property	Default value	More information
`spectrum.backup.cron`	`0 0 0 * * ?`	Quartz cron configuration for scheduled backups. Default cron schedule is every night at midnight. For more information, see **Cron Expression Generator & Explainer**.
`spectrum.backup.repository.enabled`	`true`	To specifically enable/disable backup of the neo4j repository. Uncomment this option to override the more general `spectrum.backup.enabled` setting.
`spectrum.backup.repository.databaseURL`	`127.0.0.1`	URL/Host of the machine that Neo4J is running on. Normally only change this setting if Neo4J is running on a different machine than the server. For more information, see**About scheduled backups** on page 249
`spectrum.backup.repository.port`	`6362`	The port number Neo4J is listening on for backups. Only change this when the configured port for Neo4J was modified.
`spectrum.backup.repository.directory`	`${g1.server.dir}/backup/repository`	Spectrum backup directory. Directory where backup stores files. If in a cluster, it may be advantageous to have this setting point to a network share directory. **Note:** In earlier versions the default location was *SpectrumDirectory*`/server/app/reposito`
`spectrum.backup.index.enabled`	`true`	To specifically enable or disable backup of the Elasticsearch index repository specifically. Uncomment this option to override the more general `spectrum.backup.enabled` setting.
`spectrum.backup.index.directory`	`${g1.server.dir}/backup/index`	Directory where backup will store Elasticsearch index repository files. **Important:** For a cluster, this must be set to a network share directory.

Index settings

Property	Default value	More information
spectrum.index.enabled	true	
spectrum.index.addresses	127.0.0.1	
spectrum.index.port	9200	
spectrum.index.username	admin	
spectrum.index.password	*encrypted string*	
spectrum.index.tcp.port	9300	
spectrum.index.encryption.enabled	true	
spectrum.index.encryption.keystoreType	pkcs12	
spectrum.index.encryption.keystoreAlias	spectrum	
spectrum.index.encryption.keystore	../conf/certs/spectrum-keystore.p12	
spectrum.index.encryption.keystorePassword	*encrypted string*	
spectrum.index.encryption.truststoreType	pkcs12	
spectrum.index.encryption.truststore	../conf/certs/spectrum-truststore.p12	
spectrum.index.encryption.truststorePassword	*encrypted string*	
spectrum.index.encryption.trustAllHosts	*false*	Set to true to implicitly trust all certificates; during verification, ignore host.
spectrum.index.encryption.selfSignedCerts	false	
spectrum.index.connect.timeout	180000	

Property	Default value	More information
spectrum.index.server.path	${g1.server.dir}/../index	
spectrum.index.server.encryption.keystore	../config/certs/spectrum-keystore.p12	
spectrum.index.server.encryption.truststore	../config/certs/spectrum-truststore.p12	

Debug manager settings

Property	Default value	More information
spectrum.debug.dump.dir	../exports/dumps	
spectrum.debug.dump.tmp.dir	${g1.server.tmp.dir}/dump/	
spectrum.debug.dump.maxFiles	5	
spectrum.debug.dump.crash.location	.	
spectrum.debug.dump.lib.dirs	server/deploy,server/lib	
spectrum.debug.dump.container.log.dir	server/logs	
spectrum.debug.dump.container.log.destination	server/logs	
spectrum.debug.dump.container.properties.files	server/conf/spectrum-container.properties	
spectrum.debug.dump.repository.log.dir	repository/logs	
spectrum.debug.dump.repository.log.destination	repository/logs	
spectrum.debug.dump.repository.properties.files	repository/conf/neo4j.conf	
spectrum.debug.dump.index.log.dir	index/logs	
spectrum.debug.dump.index.log.destination	index/logs	

Property	Default value	More information
spectrum.debug.dump.index.properties.files	index/config/elasticsearch.yml	

Security settings

Property	Default value	More information
spectrum.security.authentication.session.timeout	1800	Number of seconds of inactivity before client session times out (30 minutes).
spectrum.security.authentication.session.cleanup.intervalInMinutes	60	
spectrum.security.authentication.maxFailedAttempts	5	Number of failed login attempts before local account is disabled.
spectrum.security.authentication.basic.authenticator	INTERNAL	Type of authenticator used to validate *username*/*password* credentials within Spectrum. Valid values are `INTERNAL`, `LDAP`, `STS`, `SSO_STS`. For LDAP STS and SSO_STS additional configuration is needed to connect to the external authentication provider. • LDAP - /server/conf/spring/security/spectrum-config-ldap.properties • STS - /server/conf/spring/security/spectrum-config-sts.properties • SSO_STS - /server/conf/spring/security/spectrum-config-sso-sts.properties
spectrum.security.authentication.webservice.enabled.REST	true	Whether authentication is required for REST webservice calls.
spectrum.security.authentication.webservice.enabled.SOAP	true	Whether authentication is required for SOAP webservice calls.
spectrum.security.authentication.webservice.basicauth.enabled	true	Whether basic authorization is enabled for SOAP and REST webservice calls. If `false`, then only token-based authentication is available for those webservice calls.

Data manager settings

Property	Default value	More information
spectrum.data.manager.storage.dir	../ref-data	

Online help site

Property	Default value	More information
spectrum.help.url	https://lookup.docs.precisely.com	Folder containing JSP file.

Audit archive settings

Property	Default value	More information
spectrum.audit.archive.enabled	true	
spectrum.audit.archive.cron	0 0 0 ? * SUN	Default is to occur every Sunday at midnight. For more information, see **Cron Expression Generator & Explainer**.
spectrum.audit.archive.days.retain	180	Default is to retain 180 of audit data.
spectrum.audit.archive.directory	../archive/audit	Export audit data in this location before purging data.

Advanced settings

Do not modify any of these settings without contacting customer support first.

Property	Default value	More information
spectrum.remote.method.call.defaultTimeout	120000	Remote method call setting.

Property	Default value	More information
spectrum.security.account.createNonExisting	true	If using an external account repository, whether to create an instance of the user inside of Spectrum automatically if the user has been authenticated in that external repository.
spectrum.security.authentication.internal.users		Comma separated list of usernames that will be authenticated internally using Basic-Auth if authentication is configured to be external (that is, LDAP, STS). To disable all internal authentication, leave the property in place with no value. This has no effect on SSO authentication through Spectrum web applications. **Note:** Removing the property or commenting it out will not have the same behavior as leaving the property blank.
spectrum.https.encryption.excludeCipherSuites	^.*_(MD5|SHA|SHA1)$, ^TLS_RSA_.*$, ^.NULL.$, ^.anon.$, ^SSL_.*$	Comma separated regex expression for the excluded protocols. • Exclude weak / insecure ciphers. • Exclude ciphers that don't support forward secrecy. The following exclusions are present to cleanup known bad cipher suites that may be accidentally included via include patterns. • Excludes Null patterns • In case of IBM Java (AIX environment remove ^SSL_.*$ from the list): ^.*_(MD5|SHA|SHA1)$, ^TLS_RSA_.*$, ^.NULL.$, ^.anon.$
spectrum.https.encryption.includeCipherSuites	^SSL_ECDHE.*$, ^SSL_DHE.*$, SSL_RSA.*$, TLS_EMPTY_RENEGOTIATION_INFO_SCSV	Only uncomment in case of IBM JRE/JDK on AIX environment. Comma separated values for the included cipher suites only in case of AIX IBM JRE. And remove ^SSL_.*$ from the excludeCipherSuites list.

Property	Default value	More information
spectrum.log.skip.service.override	no_service_override,org.teiid	Skip override logs in service for define loggers
spectrum.security.enable.successful.login.audit	false	In case you running a high volume of web services with Spectrum, you may have a huge amount of login/logout audit events in the audit log and so overflow the audit queue. Setting this property to `false` will disable audit successful login/logouts.

Metadata settings

Property	Default value	More information
spectrum.metadata.jdbc.port	32750	By default, this TCP port number makes the JDBC connection to deployed model stores. Use this property to change the default TCP port.
spectrum.metadata.odbc.port	32751	By default, this TCP port number makes the ODBC connection to deployed model stores. Use this property to change the default ODBC port.
spectrum.job.microflow.serializer	kryo	
spectrum.job.reporting.useNumberForId	true	
g1.client.status.convert	false	
spectrum.jdbc.connectionPool.maxActive	64	
spectrum.jdbc.connectionPool.maxIdle	8	
spectrum.jdbc.connectionPool.timeBetweenEvictionRunsMillis	300000	
spectrum.jdbc.connectionPool.minEvictableIdleTimeMillis	1000000	

Property	Default value	More information
oracle.jdbc.defaultNChar	false	
spectrum.dir	../..	
spectrum.conf.dir	../conf	
spectrum.deploy.dir	../deploy	
spectrum.log.dir	../logs	
spectrum.artifacts.dir	../artifacts	
spectrum.types.dir	../types	
spectrum.resource.dir	../archive/bundles	
spectrum.modules.dir	../modules	
spectrum.archive.dir	../archive	
spectrum.microflow.dir	../tmp/microflow	
spectrum.import.dir	../import	
spectrum.exports.dir	../exports	
spectrum.jdbc.drivers.dir	../drivers	For more information, see **Importing a JDBC Driver** on page 117

Other settings

Property	Default value	More information
spectrum.server.directory.access.symlink.enable	false	Server directory access restriction property. When server directory access restriction is enabled, users can create symbolic links of directories on server and add these links to the list of accessible directories. Setting this property to `true` may create a security loophole.
spectrum.configuration.filesynch.commaseparatedfolders	../import	This property specifies the folders (comma separated) that need to be synchronized across the nodes in the cluster. This could either be absolute path or the relative path. Use `/` as the file separator.
spectrum.security.account.case.sensitive	false	Specifies whether the login username is case sensitive.

How to configure proxy server settings

If your environment uses proxy servers to provide a layer of security between clients and the Spectrum Platform Server, you will need to configure `wrapper.conf` properties to support the proxy server requests.

1. Open the following file in a code editor, such as Notepad++:
   SpectrumFolder`\server\bin\wrapper\wrapper.conf`

   Tip: You may want to back up this file before you edit it.

2. Scroll to the last parameter in the `# Java Additional Parameters` section, and note the next available parameter index.

   Tip: If the last parameter in this section is `wrapper.java.additional.27`, the next available parameter is `parameter is wrapper.java.additional.28`.

3. Add the following parameters after the last parameter in the `# Java Additional Parameters` section:

```
wrapper.java.additional.28=-Dhttp.proxyHost=HTTP proxy host IP address
wrapper.java.additional.29=-Dhttp.proxyPort=HTTP proxy port
wrapper.java.additional.30=-Dhttps.proxyHost=HTTPS proxy host IP address
wrapper.java.additional.31=-Dhttps.proxyHost=HTTPS proxy port
```

   Important: Make sure to change index value sequence (28 through 31 shown in this example) to follow the last parameter already in `wrapper.conf`.

4. Save the changes to the `wrapper.conf` file.

14 - About Spectrum Technology Platform

In this section

What Is Spectrum Technology Platform?.....542
Enterprise Data Management Architecture.....543
Spectrum Technology Platform Architecture.....547
Modules and Components.....551

What Is Spectrum Technology Platform?

Spectrum Technology Platform is a system that improves the completeness, validity, consistency, timeliness, and accuracy of your data through data standardization, verification and enhancement. Ensuring that your data is accurate, complete, and up to date enables your firm to better understand and connect with your customers.

Spectrum Technology Platform aids in the design and implementation of business rules for data quality by performing the functions described here.

Parsing, Name Standardization, and Name Validation

To perform the most accurate standardization you may need to break up strings of data into multiple fields. Spectrum Technology Platform provides advanced parsing features that enable you to parse personal names, company names, and many other terms and abbreviations. In addition, you can create your own list of custom terms to use as the basis of scan and extract operations. Spectrum Universal Name provides this functionality.

Deduplication and Consolidation

Identifying unique entities enables you to consolidate records, eliminate duplicates and develop "best-of-breed" records. A "best-of-breed" record is a composite record that is built using data from other records. Spectrum Advanced Matching and Spectrum Data Normalization provide this functionality.

Address Validation

Address validation applies rules from the appropriate postal authority to put an address into a standard form and even validate that the address is a deliverable address. Address validation can help you qualify for postal discounts and can improve the deliverability of your mail. Spectrum Universal Addressing provides this functionality.

Geocoding

Geocoding is the process of taking an address and determining its geographic coordinates (latitude and longitude). Geocoding can be used for map generation, but that is only one application. The underlying location data can help drive business decisions. Reversing the process, you can enter a geocode (a point represented by a latitude and longitude coordinate) and receive address information about the geocode. Spectrum Enterprise Geocoding provides this functionality.

Location Intelligence

Location intelligence creates new information about your data by assessing, evaluating, analyzing and modeling geographic relationships. Using location intelligence processing you can verify locations and transform information into valuable business intelligence. Spectrum Spatial provides this functionality.

Master Data Management

Master data management enables you to create relationship-centric primary data views of your critical data assets. Context Graph helps you identify influencers and non-obvious relationships, detect fraud, and improve the quality, integration, and accessibility of your information.

Tax Jurisdiction Assignment

Tax jurisdiction assignment takes an address and determines the tax jurisdictions that apply to the address's location. Assigning the most accurate tax jurisdictions can reduce financial risk and regulatory liability.

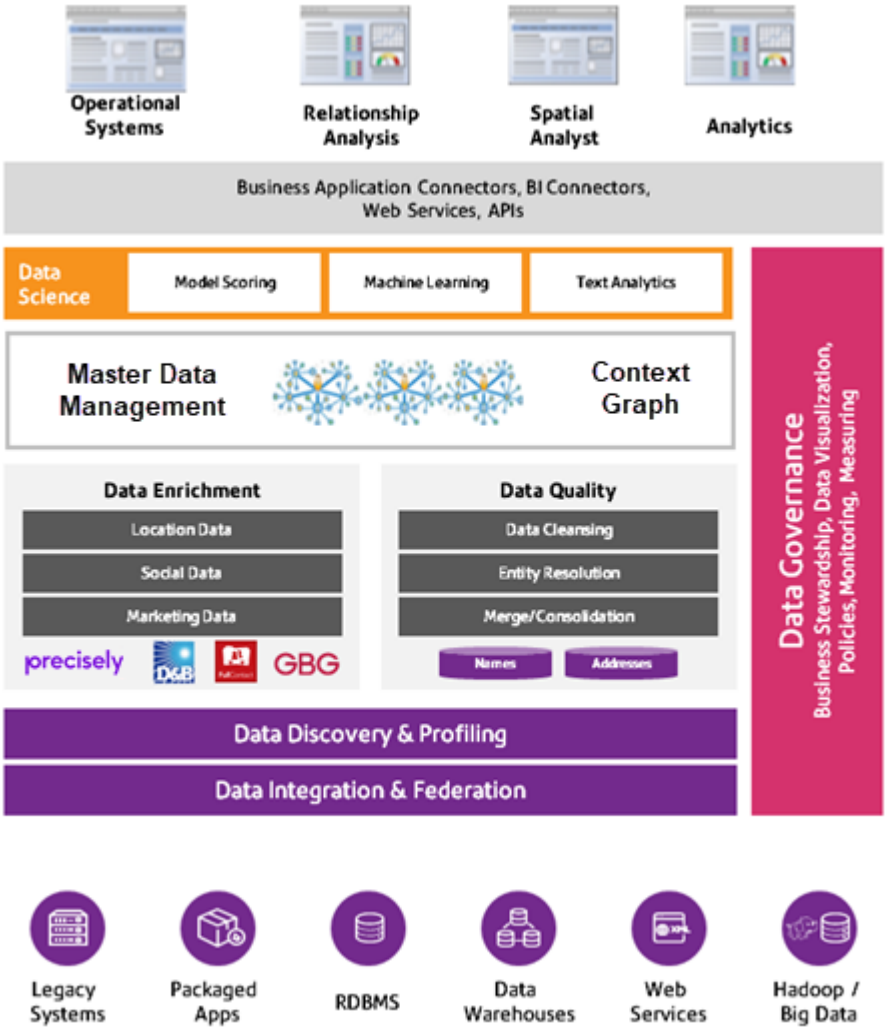
Spectrum Technology Platform software from Precisely integrates up-to-date jurisdictional boundaries with the exact street addresses of your customer records, enabling you to append the correct state, county, township, municipal, and special tax district information to your records. Some example uses of tax jurisdiction assignment are:

- Sales and use tax
- Personal property tax
- Insurance premium tax

Spectrum Enterprise Tax provides this functionality.

Enterprise Data Management Architecture

With Spectrum Technology Platform, you can build a comprehensive enterprise data management process, or you can use it as a more targeted solution. This diagram illustrates a complete solution that takes data from its source, through data enrichment and data quality processes, feeding a Master Data Management (MDM) hub which makes a single view of the data available to multiple business applications.



Master Data Management

The Master Data Management allows for rapid modeling of entities and their complex relationships across roles, processes, and interactions. It provides built-in social network analysis capabilities to help you understand influencers, predict churn, detect non-obvious relationships and fraudulent patterns, and provide recommendations.

Context Graph stores and accesses model data in a graph database. This allows users to visualize structures and relationships between entities. Users may also search the contents of models by entering natural-language queries and browsing the results. Users thereby gain insights unavailable to traditional table-based numeric analyses.

Data Enrichment

Data enrichment processes augment your data with additional information. You can base enrichment on spatial data, marketing data, or data from other detail sources. For example, if you have a database of customer addresses, you could geocode the address to determine the latitude/longitude coordinates of the address and store those coordinates as part of the record. You can then use your customer data to perform a variety of spatial calculations, such as finding the customer's nearest bank branch. Spectrum Technology Platform allows you to enrich your data with a variety of information, including geocoding (with the Spectrum Enterprise Geocoding), tax jurisdiction assignment (with Spectrum Enterprise Tax), geospatial calculations (with Spectrum Spatial), and travel directions between points (with Spectrum Spatial).

Data Quality and Data Governance

Data quality and data governance processes check your data for duplicate records, inconsistent information, and inaccurate information.

Duplicate matching identifies potential duplicate records or relationships between records, whether the data is name and address in nature or any other type of customer information. Spectrum Technology Platform allows you to specify a consistent set of business match rules using Boolean matching methods, scoring methods, thresholds, algorithms, and weights to determine if a group of records contains duplicates. Spectrum Technology Platform supports extensive customization so you can tailor the rules to the unique needs of your business.

Once duplicate records have been identified, you may wish to consolidate records. Spectrum Technology Platform allows you to specify how to link or merge duplicate records so you can create the most accurate and complete record from any collection of customer information. For example, you can build a single best-of-breed record from all of the records in a household. Spectrum Advanced Matching is used to identify duplicates and eliminate them.

Data quality processes also standardize your data. Standardization is a critical process because standardized data elements are necessary to achieve the highest possible results for matching and identifying relationships between records. While several Spectrum products perform standardization of one type or another, Spectrum Data Normalization provides the most comprehensive set of standardization features. In addition, Spectrum Universal Name provides specific data quality features for handling personal name and business name data.

Standardized data is not necessarily accurate data. Spectrum Technology Platform can compare your data to known, up-to-date reference data for correctness. The sources used for this process may include regulatory bodies such as the U.S. Postal Service, third-party data providers such as Experian or Dunn and Bradstreet, or your company's internal reference sources, such as accounting data. Spectrum Technology Platform is particularly strong in address data validation. It can validate or standardize addresses in 250 countries and territories around the world. Spectrum Universal Addressing performs address validation.

To determine which one is right for you, discuss your needs with your account executive.

While Spectrum Technology Platform can automatically handle a wide range of data quality issues, there are some situations where a manual review by a data steward is appropriate. To support this, Data Stewardship provides a way to specify the rules that will trigger a manual review, and it provides a web-enabled tool for reviewing exception records. It includes integrated access to third-party tools such as Bing maps and Experian data to aid data stewards in the review and resolution process.

Spectrum Data Discovery and Spectrum Data Profiling

Data discovery is the process of scanning your data resources to get a complete inventory of your data landscape. Spectrum Technology Platform can scan structured data, unstructured data, and semi-structured data using a wide array of data profiling techniques. The results of the scan are used to automatically generate a library of documentation describing your company's data assets and to create a metadata repository. This documentation and accompanying metadata repository provide the insight you need before beginning data integration, data quality, data governance, or master data management projects.

For more information about the Spectrum Data Discovery or Spectrum Data Profiling, contact your account executive.

Data Integration and Federation

Once you have an inventory of your data landscape, you need to consider how you will access the data you need to manage. Spectrum Technology Platform can connect to data in multiple sources either directly or through integration with your existing data access technologies. It supports batch and real-time data integration capabilities for a variety of business needs, including data warehousing, data quality, systems integration, and migration. Spectrum Technology Platform can access data in RDBMS databases, data warehouses, XML files, flat files, and more. Spectrum Technology Platform supports SQL queries with complex joins and aggregations and provides a visual query development tool. In addition, Spectrum Technology Platform can access data over REST and SOAP web services.

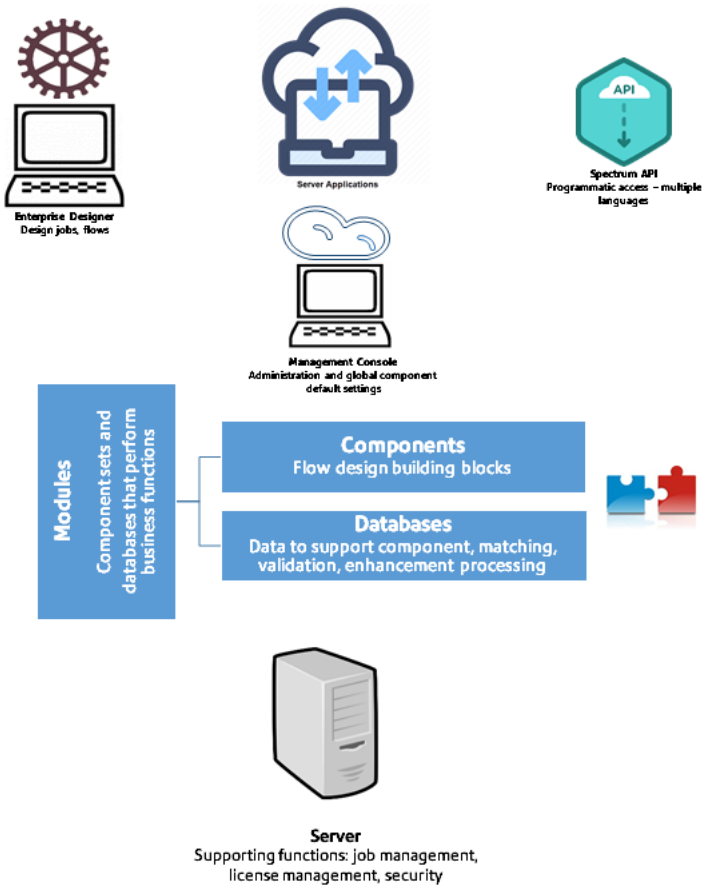
Spectrum Technology Platform can trigger batch processing based on the appearance of one or more source files in a specified folder. This "hot folder" trigger is useful for monitoring FTP uploads and processing them as they occur.

Some of these data integration capabilities require a license for the Spectrum Data Integration. For more information, contact your account executive.

Finally, Spectrum Technology Platform can integrate with packaged applications such as SAP.

Spectrum Technology Platform Architecture

Spectrum Technology Platform from Precisely consists of a server that runs a number of applications. These applications provide various functions, such as address validation, geocoding, and advanced parsing, among others. This diagram illustrates the Spectrum Technology Platform architecture.



Server

The foundation of the Spectrum Technology Platform is the server. The server handles data processing, synchronizes repository data, and manages communication. It provides job management and security features.

Products

Products are sets of features that perform a specific function. For example, Spectrum Universal Addressing standardizes addresses to conform to postal standards. Spectrum Enterprise Tax

determines the tax jurisdictions that apply to a given address. Context Graph provides graph models and data visualization. Products are grouped together to solve common business problems and licensed together in bundles.

Components

Products are comprised of components which provide applications and perform specific functions in a flow or as a service. For example, the Geocode US Address component in Spectrum Enterprise Geocoding takes an address and returns the latitude and longitude coordinates for that address; Get City State Province in Spectrum Universal Addressing takes a postal code and returns the city and state or province where that postal code is located.

The components that you have available on your system depend on which Spectrum Technology Platform bundle you have licensed.

Databases

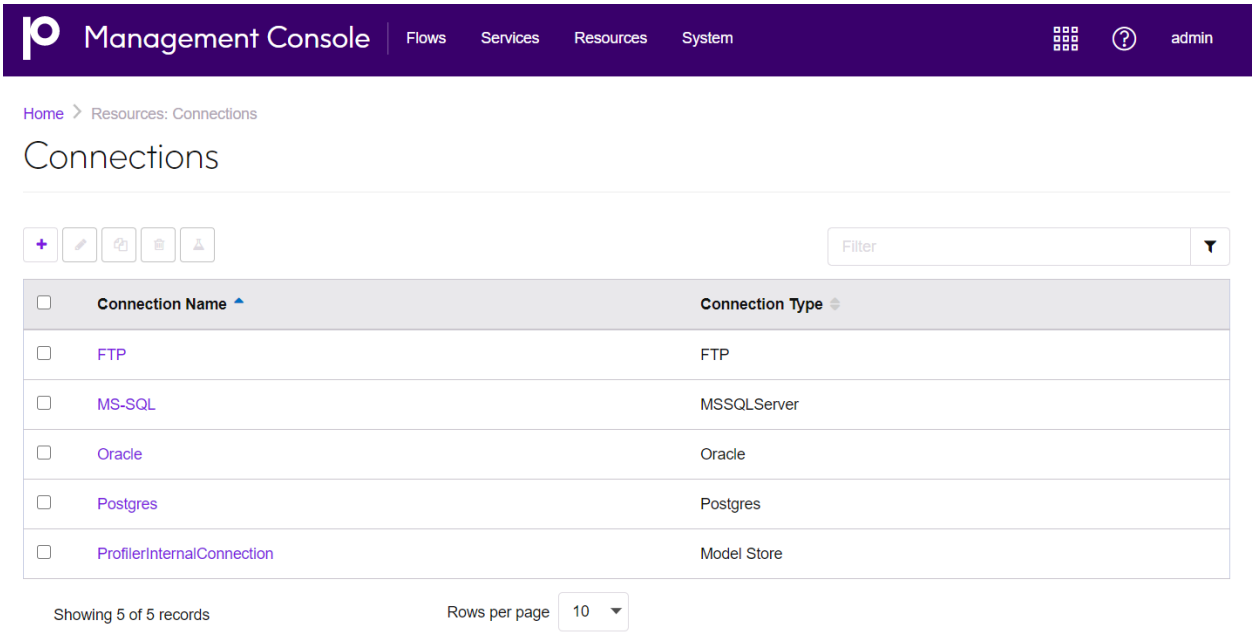
Some products depend on databases containing reference data. For example, Spectrum Universal Addressing needs to have access to U.S. Postal Service data in order to verify and standardize addresses in the U.S. Databases are installed separately and some are updated on a regular basis to provide you with the latest data. Context Graph, on the other hand, stores graph models in an external Neo4j graph database server.

In addition to required databases, have optional databases. Optional databases provide data needed for certain features that can enhance your Spectrum Technology Platform process.

Spectrum Management Console

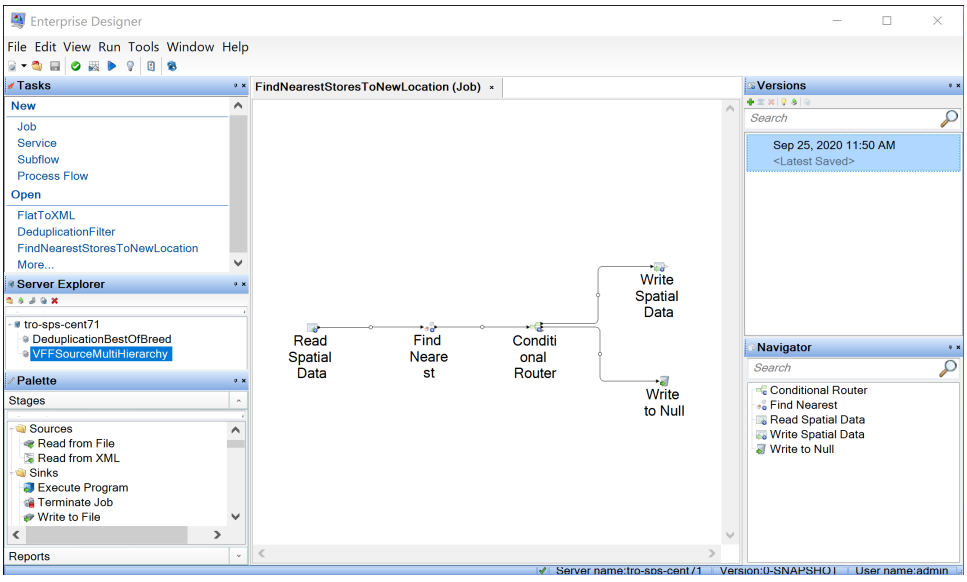
Spectrum Management Console is a tool for administering Spectrum Technology Platform. You can use Spectrum Management Console to:

- Define the connections between Spectrum Technology Platform and your data
- Specify the default settings for services and flows
- Manage user accounts, including permissions and passwords
- View logs
- View licenses including license expiration information



Spectrum Enterprise Designer

Spectrum Enterprise Designer is a tool for creating Spectrum Technology Platform jobs, services, subflows, and process flows. It provides a familiar drag-and-drop interface to allow you to graphically create complex flows.



Discovery

Spectrum Discovery gives you the control you need to deliver accurate and timely data-driven insights to your business. Use Spectrum Discovery to develop data models, view the flow of data from source

to business application, and assess the quality of your data through profiling. With this insight, you can identify the data resources to use to answer particular business questions, adapt and optimize processes to improve the usefulness and consistency of data across your business, and troubleshoot data issues.

Web Services and API

You can integrate Spectrum Technology Platform capabilities into your applications using web services and programming APIs. These interfaces provide simple integration, streamline record processing, and support backward compatibility of future versions.

The Spectrum Technology Platform API is available for these languages:

- C
- C++
- COM
- Java
- .NET

Web services are available via SOAP and REST.

Spectrum Administration Utility - Command Line Interface (CLI)

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

Modules and Components

Table 3: Modules and Components

Module	Description	Components
Spectrum Advanced Matching	Matches records within or between input files.	Best Of Breed Candidate Finder Duplicate Synchronization Filter Interflow Match Intraflow Match Match Key Generator Transactional Match
Spectrum Data Stewardship	Identifies exception records and provides a browser-enabled tool for manually reviewing exception records.	Exception Monitor Read Exceptions Write Exceptions
Country Identifier	Takes a country name or a combination of postal code and state-province and returns the two-character ISO country code, the three-character Universal Postal Union (UPU) code, and the English country name.	Country Identifier
Spectrum Discovery	Gives you the control you need to deliver accurate and timely data-driven insights to your business. Develops data and graph models, gives you a view the flow of data from source to business application, and assesses the quality of your data through profiling. It helps you identify the data resources you should use to answer particular business questions and to optimize processes to improve the usefulness and consistency of data across your business.	Models (Logical, Physical, and Context Graph) Model Store Profile Lineage & Impact Analysis

Module	Description	Components
Spectrum Context Graph	Links and analyzes data, identifying relationships and trends.	Write to Model Read From Model Query Model Graph Visualization
Spectrum Data Federation	Provides capabilities useful in data warehousing, data quality, systems integration, and migration.	Field Selector Generate Time Dimension Query Cache Write to Cache
Spectrum Data Normalization	Removes inconsistencies in data.	Advanced Transformer Open Parser Table Lookup Transliterator
Spectrum Data Integration	Connects to data in multiple sources for a variety of business needs including data warehousing, data quality, systems integration, and migration.	Call Stored Procedure Field Selector Generate Time Dimension Query Cache Write to Cache
Spectrum Geocoding	Determines the geographic coordinates for an address. Also determines the address of a given latitude and longitude.	Geocode Address AUS Geocode Address GBR - deprecated. Use Global Geocoding geocoding stage. Geocode Address Global Geocode Address World Geocode US Address GNAF PID Location Search Reverse APN Lookup Reverse Geocode Address Global Reverse Geocode US Location

Module	Description	Components
Spectrum Enterprise Tax	Determines the tax jurisdictions that apply to a given location.	Assign GeoTAX Info Calculate Distance
GeoConfidence	Determines the probability that an address or street intersection is within a given area.	Geo Confidence Surface CreatePointsConvexHull
Spectrum Global Addressing	Provides enhanced address standardization and validation. Also, automatically suggests addresses as you type and immediately returns candidates based on your input. Splits postal address strings into individual address elements using machine learning techniques.	Global Address Validation Global Type Ahead
Spectrum Global Geocoding	Determines the geographic coordinates for an address. Also determines the address of a given latitude and longitude. Interactive geocoding is a type-ahead feature in Global Geocoding. Key Lookup uses a key to geocode addresses.	Global Geocode Global Reverse Geocode Spectrum Global Interactive Geocoding Global Key Lookup
Spectrum Global Sentry	Attempts to match transactions against government-provided watch lists that contain data from different countries.	Global Sentry Global Sentry Address Check Global Sentry ID Number Check Global Sentry Name Check Global Sentry Other Data Check

Module	Description	Components
Spectrum Spatial	Performs point in polygon and radial analysis against a variety of geospatial databases.	Find Nearest Point In Polygon Query Spatial Data Read Spatial Data Spatial Calculator Spatial Union Write Spatial Data
	Performs routing calculations to obtain directions, calculate drive time and drive distance, and identify locations within a certain time or distance from a starting point.	Matrix Route Travel Boundary
Spectrum SAP	Enables Spectrum Technology Platform to interface with SAP Customer Relationship Management applications.	SAP Generate Match Key SAP Generate Match Score SAP Generate Search Key SAP Generate Search Key Constant SAP Generate Search Key Metaphone SAP Generate Search Key Substring SAP Validate Address With Candidates
Spectrum Universal Address	Standardizes and validates addresses according to the postal authority's standards.	Get Candidate Addresses Get City State Province Get Postal Codes Validate Address
Spectrum Universal Name	Parses personal names, company names, addresses, and many other terms and abbreviations.	Name Parser (Deprecated) Name Variant Finder Open Name Parser

Notices

Information in this document is subject to change without notice and does not represent a commitment on the part of the vendor or its representatives. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, without the written permission of Precisely, 1700 District Ave Ste 300, Burlington MA 01803-5231, USA.

Copyright 2007, 2022 Precisely

See **www.precisely.com** for information about our trademarks.

USPS® Notices

Precisely holds a non-exclusive license to publish and sell ZIP + 4® databases on optical and magnetic media. These trademarks are owned by the United States Postal Service: CASS, CASS Certified, DPV, eLOT, FASTforward, First-Class Mail, Intelligent Mail, LACS^{Link}, NCOA^{Link}, PAVE, PLANET Code, Postal Service, POSTNET, Post Office, RDI, Suite^{Link}, United States Postal Service, Standard Mail, United States Post Office, USPS, ZIP Code, and ZIP + 4. This list is not exhaustive of the trademarks belonging to the Postal Service.

Precisely is a non-exclusive licensee of USPS® for NCOA$^{Link®}$ processing.

Prices for Precisely products, options, and services are not established, controlled, or approved by USPS® or United States Government. When utilizing RDI™ data to determine parcel-shipping costs, the business decision on which parcel delivery company to use is not made by the USPS® or United States Government.

Data Provider and Related Notices

Data Products used within Precisely applications are protected by various trademarks and by one or more of these copyrights:

© 2022 United States Postal Service. All rights reserved.

© 2022 TomTom. TomTom. All rights reserved. This material is proprietary and the subject of copyright protection and other intellectual property rights owned by or licensed to TomTom or its suppliers. The use of this material is subject to the terms of a license agreement. Any unauthorized copying or disclosure of this material will lead to criminal and civil liabilities.

© 2022 HERE Fuente: INEGI (Instituto Nacional de Estadística y Geografía) - Based upon electronic data © National Land Survey Sweden.

© 2012 United States Census Bureau

Portions of this program are © Copyright 1993-2022 by Nova Marketing Group Inc. All Rights Reserved

© 2022 Second Decimal, LLC

© 2022 Canada Post Corporation - Data is from a compilation in which Canada Post Corporation is the copyright owner.

© 2022 Claritas, Inc.

The Geocode Address World data set contains data licensed from the GeoNames Project (**www.geonames.org**) provided under the Creative Commons Attribution License ("Attribution License") located at **http://creativecommons.org/licenses/by/3.0/legalcode**. Your use of the GeoNames data (described in the Spectrum Technology Platform User Manual) is governed by the terms of the Attribution License, and any conflict between your agreement with Precisely and the Attribution License will be resolved in favor of the Attribution License solely as it relates to your use of the GeoNames data.

precisely

1700 District Ave Ste 300
Burlington MA 01803-5231
USA

www.precisely.com

Copyright 2007, 2022 Precisely