

precisely

Spectrum Enterprise Designer

Spectrum Dataflow Designer's Guide

Version 2020.1.0



Table of Contents

1 - Getting Started

Installing the Client Tools.....	5
Starting Spectrum Enterprise Designer.....	6
A First Look at Spectrum Enterprise Designer.....	6
My First Dataflow (Job).....	9
My First Dataflow (Service).....	12
Dataflow Templates.....	14
Importing and Exporting Dataflows.....	15

2 - Designing a Flow

Types of Flows.....	18
Flow Input.....	20
Fields.....	25
Control Stages.....	39
Module Stages.....	96
Flow Output.....	783
Embedded flows.....	788
Reports.....	792
Performance Considerations.....	797
Flow Versions.....	810

3 - Inspecting and Testing

Checking a Flow for Errors.....	815
Inspecting a flow.....	815
Testing a service with Spectrum Management Console.....	819

4 - Running a Flow

Running a Job or Process Flow.....	821
Exposing a Service.....	841

Runtime Options.....	843
Configuring Email Notification for a Flow.....	846

5 - Combining Flows into a Process Flow

Introduction to Process Flows.....	849
Designing Process Flows.....	849

6 - Creating Reusable Flow Components

Introduction to Subflows.....	863
Using a Subflow as a Source.....	863
Using a Subflow in the Middle of a Flow.....	864
Using a Subflow as a Sink.....	865
Modifying a Subflow.....	866
Deleting a Subflow.....	867
Exposing and Unexposing a Subflow.....	867
Converting a Stage to a Subflow.....	867

7 - Sample Flows

Introduction.....	870
Integration between SugarCRM OnPremises and Microsoft Dynamics 365 Online.....	872
Integration between Salesforce and Oracle Eloqua.....	874

8 - About Spectrum Technology Platform

What Is Spectrum Technology Platform?.....	878
--	-----

Enterprise Data Management Architecture.....879
Spectrum Technology Platform Architecture.....883
Modules and Components.....887

1 - Getting Started

In this section

Installing the Client Tools.....	5
Starting Spectrum Enterprise Designer.....	6
A First Look at Spectrum Enterprise Designer.....	6
My First Dataflow (Job).....	9
My First Dataflow (Service).....	12
Dataflow Templates.....	14
Importing and Exporting Dataflows.....	15



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.
- **Flow Designer** is the next-generation Web UI dataflow design tool. This release provides a technical *preview* version of Flow Designer.

Note: Enterprise Designer will be retired once Flow Designer contains the full feature set in a future release.

- **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**.
3. Download the client tool you want to install.

Starting Spectrum Enterprise Designer

Spectrum Enterprise Designer is a Windows application for creating dataflows. To start Spectrum Enterprise Designer:

1. Select **Start > Precisely > Spectrum Enterprise Designer**.
2. Enter the server name or IP address, or select it from the drop-down list. If you are using Spectrum Technology Platform in a cluster, enter the name of IP address of the cluster's load balancer.
3. Enter your user name and password.
4. In the Port field, enter the network port that the server has been configured to use for Spectrum Technology Platform communication. The default port number is 8080.
5. Click **Use secure connection** if you want communication between the client and the server to take place over an HTTPS connection.

Note: A secure connection is only available if HTTPS communication has been configured on the server. If you are running Spectrum Enterprise Designer on Windows 7, using the IP address in the **Server name** field may not work, depending on the type of certificate used to secure the communication between Spectrum Enterprise Designer and the server. If the IP address does not work, use the host name instead.

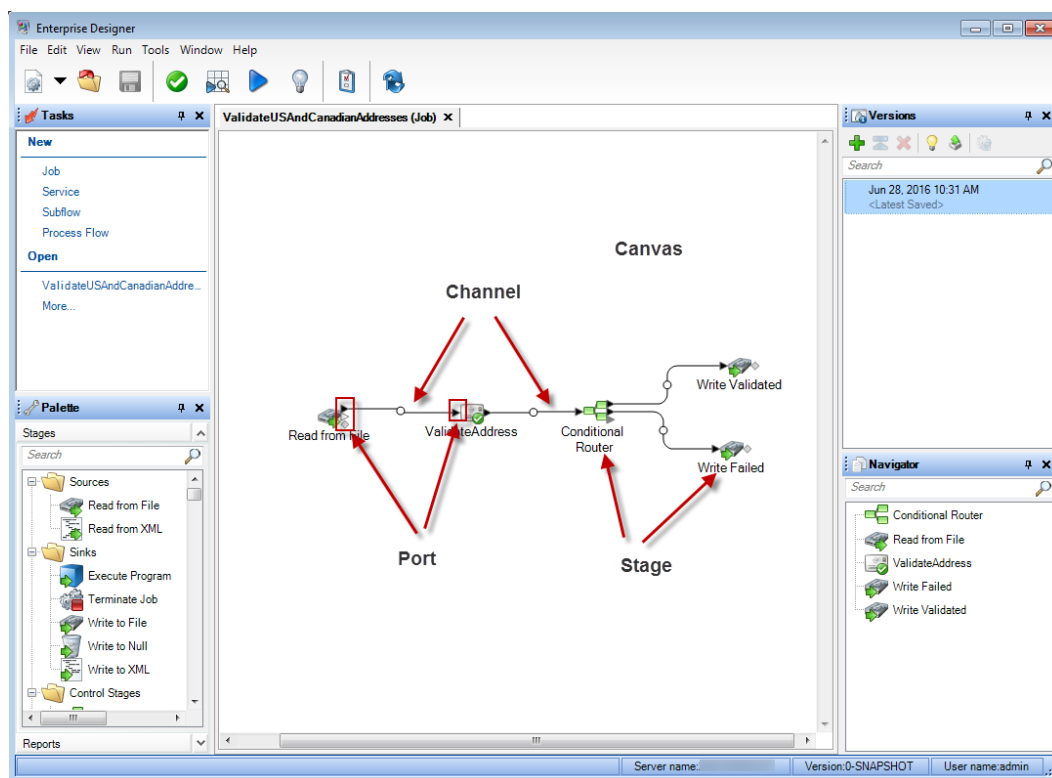
6. Click **Login**.

A First Look at Spectrum Enterprise Designer

Spectrum Enterprise Designer is a visual tool for creating dataflows. Using this client, you can:

- Create and modify jobs, services, subflows, and process flows
- Inspect and validate dataflows for correctness
- Expose and hide services
- Generate reports

The Spectrum Enterprise Designer window looks like this:



These concepts are important for working with flows:

Canvas The canvas is the main work area. The picture above shows the canvas open with a dataflow named `ValidateUSAndCanadianAddresses`. It is a job dataflow, which means it performs batch processing by reading data from a file and writing output to a file. In this case, the dataflow is writing output to two files.

Stage Stages, represented by icons on the canvas, perform a specific type of activity, such as sorting records, validating addresses, matching similar records, and so on. To add a stage, drag the stage from the Palette (on the left side of the window) onto the canvas.

If a stage requires your attention, a blue circle appears on the icon:



A dataflow cannot run successfully if it has stages that require attention. So, double-click the stage to configure the required settings. Once you have configured all the required settings, the blue circle no longer appears:



Channel A channel is a connection between two or more stages through which records are passed from one stage to another. In the above example, you can see that the `Read from File` stage is connected to the `ValidateAddress` stage with a channel. Records are read into

the dataflow in Read from File then sent to ValidateAddress through this channel. ValidateAddress is then connected to Conditional Router through a channel. Conditional Router, which analyzes records and sends them along different paths in a dataflow depending on the conditions defined by the dataflow designer, has two channels going out of it, one to a Write Validated stage and one to a Write Failed stage.

The dot in the middle of a channel may change colors to indicate different conditions:

Red	Indicates an error, such as a type conversion failure that makes a field unusable by the downstream stage.
Yellow	You have removed a field that is needed by a downstream stage.
Blue	Automatic type conversion has successfully converted a field to the data type required by the downstream stage.
Black	A field is being renamed in the channel.
White	No action is being taken on fields.

Port If you look closely at the stage icons you will notice small triangular or diamond shaped ports on the sides of each stage. A port is the mechanism by which a stage sends data into, or reads data from, a channel. Stages that read data into the dataflow (called "sources") only have output ports since they are always at the start of a dataflow. Stages that send data out of the dataflow (called "sinks") only have input ports since they are always at the end of a dataflow. All other stages have both input and output ports. In addition, some stages have error ports, which are used to output records that cause errors during the stage's processing, and some stages have report ports, which are used to generate reports about the stage's output.

The Spectrum Enterprise Designer window provides these features:

Feature	Description
Tasks	Provides a quick way to create a new job, service, subflow, or process flow. Also allows you to open dataflows that were recently open.
Server Explorer	Shows all the flows saved on the Spectrum Technology Platform server. If the server explorer this is not visible, select View > Server Explorer . You can organize flows into folders. To create a folder, right-click the server name and select New Folder . Flow names must be unique across all folders. You cannot have two flows with the same name even if they are in different folders.

Feature	Description
Palette	Contains all the stages and reports you can add to your dataflow. The stages available in the palette depend on the modules you have licensed.
Canvas	The work area onto which you drag stages and connect them with channels to make dataflows. You can have several dataflow canvases open at once.
Versions	The Versions feature in Spectrum Enterprise Designer allows you to keep a revision history of your flows. You can view previous versions of a flow, expose older versions for execution, and keep a history of your changes in case you ever need to revert to a previous version of a flow.
Navigator	Lists the stages and reports in the flow. You can right-click an item in the Navigator pane to edit its options.

My First Dataflow (Job)

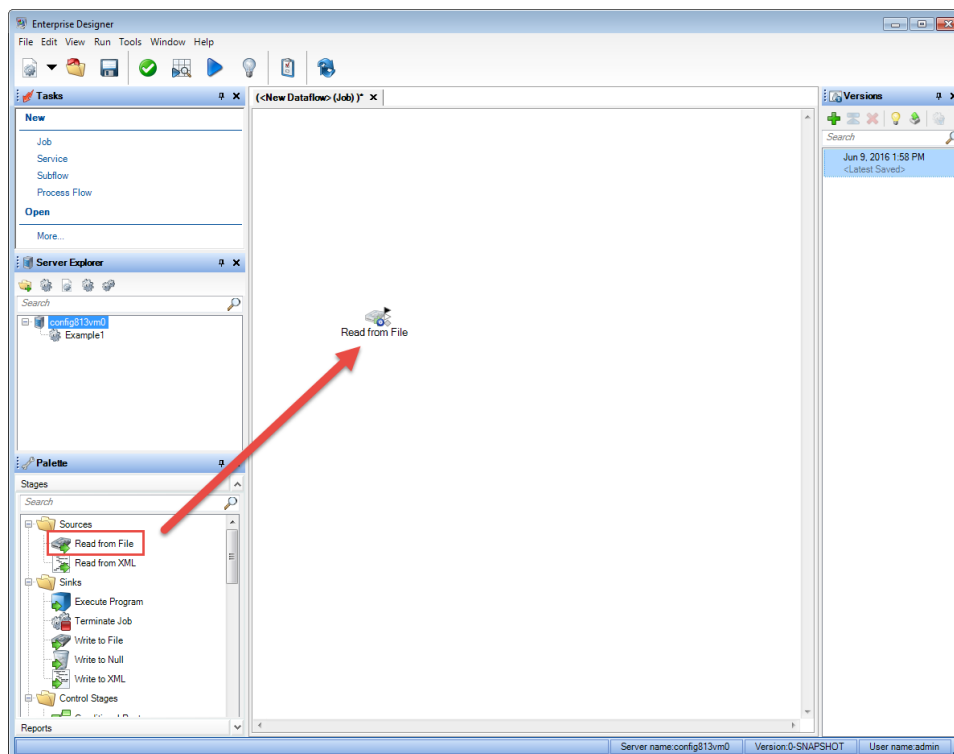
This example shows how to create a simple dataflow that reads data from a file, sorts it, then writes it to a file. Since this dataflow reads data from a file and writes its output to a file, it is a **job**: a dataflow that performs batch processing. (The other primary type of dataflow, a **service**, performs interactive processing via an API or web service call to the server.)

1. The first step will be to create some sample data to use as input to your dataflow. Using a text editor, create a file that looks like this:

```
FirstName,LastName,Region,Amount
Alan,Smith,East,18.23
Jeannie,Wagner,North,45.43
Joe,Simmons,East,10.87
Pam,Hiznay,Central,98.78
```

2. Save the file in a convenient location.
3. Select **Start > Programs > Precisely > Spectrum Technology Platform > Client Tools > Enterprise Designer**.
4. Select **File > New > Dataflow > Job**.
5. You are now ready to begin creating your dataflow. The first step is to define the input to the dataflow. To do this:

- a) Drag a Read from File stage to the canvas:

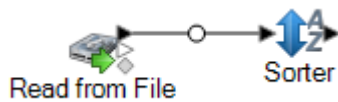


- b) Double-click the Read from File stage on the canvas.
 c) In the **File name** field, specify the file you created for this task.
 d) In the **Record type** field, choose **Delimited**.
 e) In the **Field separator** field, select **Comma (,)**.
 f) Check the **First row is header record** box.
 g) Click the **Fields** tab.
 h) Click **Regenerate** then click **Yes**.

The stage is automatically configured for the fields in your input file.

- i) Click **Detect Type**. This scans the input file and determines the appropriate data type for each field. Notice that the type for the **Amount** field changes from string to double.
 j) You have finished configuring Read from File. Click **OK**.
6. Next, you will add a stage that will sort the records by region. To do this:
- a) Drag the Sorter stage to the canvas
 b) Click the solid black triangle on the right side of the Read from File stage (the output port) and drag it to the left side of the Sorter stage on the canvas to create a channel connecting Read from File and Sorter.

Your dataflow should look like this:



- c) Double-click the Sorter stage on the canvas.
 - d) Click **Add**.
 - e) In the **Field Name** field, select **Region**.
 - f) You have finished configuring Sorter. Click **OK**.
7. Finally, you will define the output file where the dataflow will write its output. To do this:
- a) Drag a Write to File stage to the canvas.
 - b) Click the solid black triangle on the right side of the Sorter stage and drag it to the left side of the Write to File stage on the canvas.

Your dataflow should look like this:



- c) Double-click the Write to File stage.
- d) In the **File name** field, specify an output file. This can be any file you want.
- e) In the **Field separator** field, select **Comma (,)**.
- f) Check the **First row is header record** box.
- g) Click the **Fields** tab.
- h) Click **Quick Add**.
- i) Click **Select All** then click **OK**.
- j) Using the **Move Up** and **Move Down** buttons, reorder the fields so that they are in the following order:

FirstName
 LastName
 Region
 Amount

This will make the records in your output file have the fields in the same order as your input file.

- k) You have finished configuring Write to File. Click **OK**.
8. In Enterprise Designer, select **File > Save**.
 9. Give your dataflow a name and click **OK**.
 10. Your dataflow is now ready to run. Select **Run > Run Current Flow**.
 11. The **Execution Details** window appears and shows the status of the job. Click **Refresh**. Once the status shows **Succeeded** click **Close**.

Open the output file you specified in the Write to File stage. You will see that the records have been sorted by region as you specified in the Sorter stage.

```

FirstName, LastName, Region, Amount
Pam, Hiznay, Central, 98.78
Alan, Smith, East, 18.23
Joe, Simmons, East, 10.87
Jeannie, Wagner, North, 45.43

```

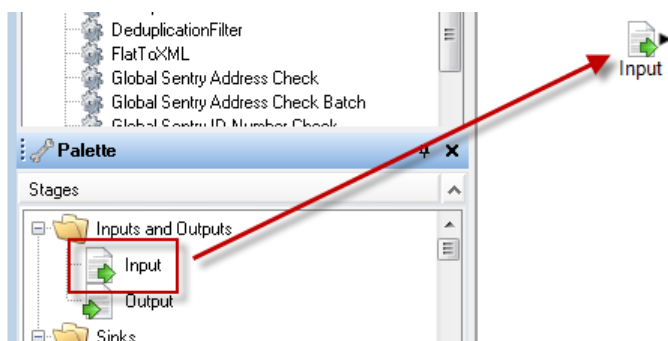
Congratulations! You have designed and run your first job dataflow.

My First Dataflow (Service)

This example shows how to create a simple dataflow that accepts data from an API or web service call, processes the data, and returns a response via the API or web service. Since this dataflow is intended to be exposed as a service on the Spectrum Technology Platform server, it is a **service** dataflow. (The other primary type of dataflow, a **job**, performs batch processing, reading data from a file or database, processing the data, then writing the output to a file or database.)

1. Select **Start > Programs > Precisely > Spectrum Technology Platform > Client Tools > Enterprise Designer**.
2. Select **File > New > Dataflow > Service**.
You are now ready to begin creating your dataflow.
3. The first step is to define the input to the dataflow.
Your dataflow will take two fields as input: FirstName and LastName.

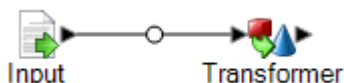
a) Drag an Input stage from the palette to the canvas.



- b) Double-click the Input stage on the canvas.
- c) Click **Add** then click **Add** again.
- d) In the **Field name** field, type `FirstName`.
- e) Click **OK**, then click **OK** again.
- f) Click **Add** then click **Add** again.
- g) In the **Field name** field, type `LastName`.

- h) Click **OK**, then click **OK** again.
 - i) You have finished defining the dataflow input. Click **OK**.
4. Next, you will add a stage to change the casing of the data in the FirstName and LastName fields to all upper case.
- a) Drag a Transformer stage from the palette to the canvas.
 - b) Click the solid black triangle on the right side of the Input stage (the output port) and drag it to the left side of the Transformer stage on the canvas to create a channel connecting Input and Transformer.

Your dataflow should look like this:



- c) Double-click the Transformer stage.
 - d) Click **Add**.
 - e) In the tree on the left side, under **Formatting** click **Case**.
 - f) In the **Field** field, select **FirstName**. Leave **Upper** selected.
 - g) Click **Add**.
 - h) In the **Field** field, select **LastName**. Leave **Upper** selected.
 - i) Click **Add**.
 - j) Click **Close**.
 - k) You have finished configuring Transformer to change the value in the FirstName and LastName fields to upper case. Click **OK**.
5. Finally, you will define the output for the dataflow. Your dataflow will return the FirstName and LastName fields as output.
- a) Drag an Output stage to the canvas.
 - b) Click the solid black triangle on the right side of the Transformer stage and drag it to the left side of the Output stage on the canvas.

Your dataflow should look like this:



- c) Double-click the Output stage on the canvas.
 - d) Check the **Expose** box. The check boxes next to FirstName and LastName should now be checked.
 - e) Click **OK**.
6. In Enterprise Designer, select **File > Save**.
7. Give your dataflow the name `MyFirstDataflow-Service` and click **OK**.
8. Select **File > Expose/Unexpose and Save**. This exposes your dataflow, making it available as a service on the server.


9. To test your service:

a) Open Management Console by going to this URL in a web browser:

```
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) Go to **Services > Other Services**.

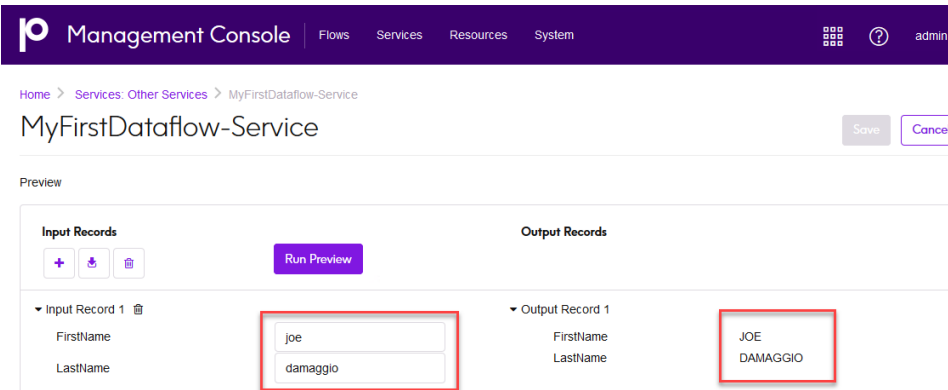
c) In the list of services, check the box next to **MyFirstDataflow-Service** then click the Edit button .

d) Enter a name in the FirstName field in all lower case letters.

e) Enter a name in the LastName field in all lower case letters.

f) Click **Run Preview**.

You can see that the service made the name fields all upper case letters, as you specified in your dataflow Transformer stage.



The screenshot shows the Management Console interface. At the top, there is a navigation bar with 'Management Console' and tabs for 'Flows', 'Services', 'Resources', and 'System'. The breadcrumb trail is 'Home > Services: Other Services > MyFirstDataflow-Service'. The main heading is 'MyFirstDataflow-Service' with 'Save' and 'Cancel' buttons. Below this is a 'Preview' section. It contains two columns: 'Input Records' and 'Output Records'. In the 'Input Records' column, there is a table with one record: 'Input Record 1' containing 'joe' for 'FirstName' and 'damaggio' for 'LastName'. In the 'Output Records' column, there is a table with one record: 'Output Record 1' containing 'JOE' for 'FirstName' and 'DAMAGGIO' for 'LastName'. A 'Run Preview' button is located between the two columns. Red boxes highlight the input and output fields to show the transformation from lowercase to uppercase.

Congratulations! You have designed and run your first service dataflow. The service is now available on the server and can be accessed via an API or web services call. The resource URL for this service's SOAP endpoint is:

```
http://<ServerName>:<Port>/soap/MyFirstDataflow-Service
```

The resource URL for this service's REST endpoint is:

```
http://<ServerName>:<Port>/rest/MyFirstDataflow-Service
```

Dataflow Templates

Dataflow templates illustrate ways in which you can use Spectrum Technology Platform and its modules to meet your business needs. They show how particular modules solve various requirements,

such as parsing, standardizing, and validating names and addresses, geocoding addresses, and so on.

Dataflow templates are delivered with each module that you license. For instance, if you are licensed for Spectrum Data Normalization, you receive the Standardizing Personal Names dataflow template. If you are licensed for Spectrum Universal Addressing, you receive the Validating U.S. and Canadian Addresses dataflow templates.

Depending on the purpose of each template, it may be a job with sample data or it may be a service with no sample data. You can use dataflows in their original state and run those that are delivered as jobs to see how they function. Alternatively, you can manipulate the dataflows by changing input and output files or by bringing services into your own jobs and adding input and output files.

Note: These samples are intended as illustrations of various Spectrum Technology Platform features. They are intended to be starting points and examples for solutions you can create for your environment.

Creating a Dataflow Using a Template

Dataflow templates are delivered with each module that you license. To create a dataflow using a template,

- In Enterprise Designer go to **File > New > Dataflow > From Template**.
- Or, you can click the **New** icon and select New Dataflow From Template.

A list of templates available for the modules you have installed is displayed.

Importing and Exporting Dataflows

You can exchange dataflows with other Spectrum Enterprise Designer users with the import and export features.

Note: Dataflows can only be exchanged between identical versions of Spectrum Technology Platform.

- To export a dataflow, select **File > Export**. If you have used the Versions feature to save versions of the dataflow, the version you have currently selected is the version that is exported.

Note: Do not use special characters in the name of the services and jobs you define. Doing so may result in an error during export.

- To import a process flow, select **File > Import > Process Flow**.

- To import a dataflow, select **File > Import > Dataflow**. The stages in the dataflow must be available on your system before you import the dataflow. If the dataflow you import contains unavailable stages, you will see an error.
- If you use Server Explorer to organize your dataflows you can also export a dataflow by right-clicking it and selecting **Export**. To import a dataflow using Server Explorer, right-click in the location in Server Explorer where you want to import the dataflow and select **Import**.

2 - Designing a Flow

In this section

Types of Flows.....	18
Flow Input.....	20
Fields.....	25
Control Stages.....	39
Module Stages.....	96
Flow Output.....	783
Embedded flows.....	788
Reports.....	792
Performance Considerations.....	797
Flow Versions.....	810



Types of Flows

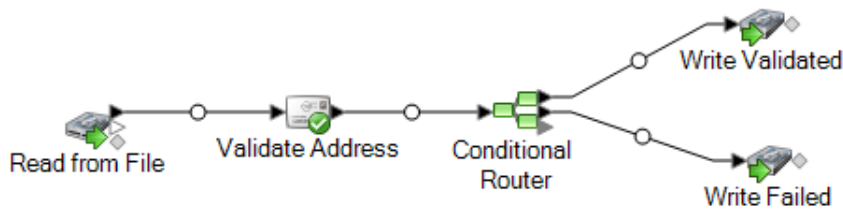
A dataflow is a series of operations that takes data from some source, processes that data, then writes the output to some destination. The processing of the data can be anything from simple sorting to more complex data quality and enrichment actions. The concept of a dataflow is simple, but you can design very complex dataflows with branching paths, multiple sources of input, and multiple output destinations.

There are four types of dataflows: jobs, services, subflows, and process flows.

Job

A job is a dataflow that performs batch processing. A job reads data from one or more files or databases, processes that data, and writes the output to one or more files or databases. Jobs run manually through the UI or from a command line using the job executor.

This dataflow is a job. Note that it uses the Read from File stage for input and two Write to File stages as output.

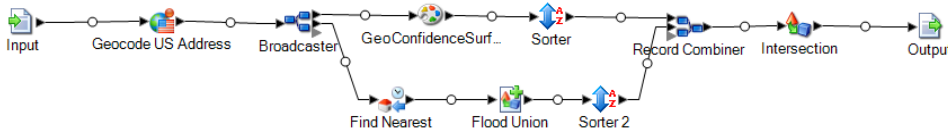


Service

A service is a dataflow that you can access as web services or using the Spectrum Technology Platform API. You pass a record 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.

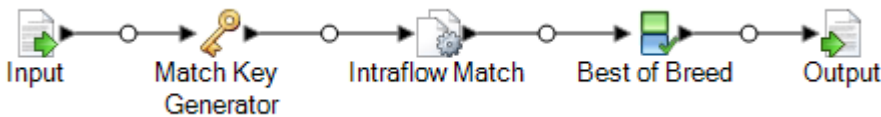
You can also design your own custom services in Spectrum Enterprise Designer. For example, the following dataflow determines if an address is at risk for flooding:



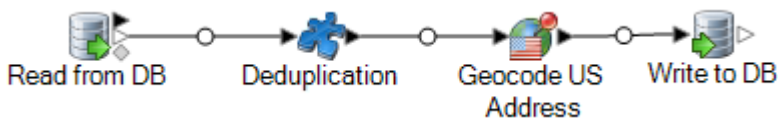
Note: Since the service name, option name, and field name ultimately become XML elements, they may not contain characters that are invalid in XML element names (for example, spaces are not valid). Services not meeting this requirement will still function but will not be exposed as web services.

Subflow

A subflow is a dataflow that can be reused within other dataflows. Subflows are useful when you want to create a reusable process that can be easily incorporated into dataflows. For example, you might want to create a subflow that performs deduplication using certain settings in each stage so that you can use the same deduplication process in multiple dataflows. To do this you could create a subflow like this:



You could then use this subflow in a dataflow. For example, you could use the deduplication subflow within a dataflow that performs geocoding so that the data is deduplicated before the geocoding operation:

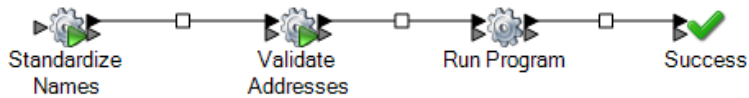


In this example, data would be read in from a database then passed to the deduplication subflow, where it would be processed through Match Key Generator, then Intraflow Match, then Best of Breed, and finally sent out of the subflow and on to the next stage in the parent dataflow, in this case Geocode US Address. Subflows are represented as a puzzle piece icon in the dataflow, as shown above.

Subflows that are saved and exposed are displayed in the **User Defined Stages** folder.

Process Flow

A process flow runs a series of activities such as jobs and external applications. Each activity in the process flow runs after the previous activity finishes. Process flows are useful if you want to run multiple flows in sequence or if you want to run an external program. For example, a process flow could run a job to standardize names, validate addresses, then invoke an external application to sort the records into the proper sequence to claim postal discounts. Such a process flow would look like this:



In this example, the jobs Standardize Names and Validate Addresses are exposed jobs on the Spectrum Technology Platform server. Run Program invokes an external application, and the Success activity indicates the end of the process flow.

Flow Input

To define the input for a dataflow, use a **source** stage. A source is the first stage in a dataflow. It defines the input data you want to process.

Input for a Job

Input data for a job can come from a file or a database. Spectrum Technology Platform has the ability to read data from many file formats and database types. The types of data sources you can read from depend on which Spectrum processes you have licensed. Spectrum Data Integration provides access to the most data sources of any module.

Note: When designing a job, it is a good idea to account for the possibility of malformed input records. A malformed record is one that cannot be parsed using one of the parser classes provided by Spectrum Technology Platform. For information about handling malformed input records, see [Managing malformed input records](#) on page 21.

Input for a Service

Input data for a service is defined in an Input stage. This stage defines the fields that the service will accept from a web service request or an API call.

Defining Job Input

Input data for a job can come from a file, database, or cloud service, depending on the modules you have licensed. Each module supports input from different sources, and the procedure for configuring each type of source varies greatly. See the solution guide for your modules available at support.precisely.com.

Managing malformed input records

A malformed record is one that Spectrum Technology Platform cannot parse. When Spectrum Technology Platform encounters a malformed record, it can do one or more of these tasks:

- Terminate the job
- Continue processing
- Continue processing until a certain number of bad records are encountered
- Continue processing but write bad records to a log file (via an optional sink stage)

Note: Malformed records functionality is limited to sources configured to read from files local to the server and that do not have sorting configured. When a source is configured with either a remote file or with sort fields and the source encounters a malformed record, the job will terminate regardless of the configuration for malformed records.

To manage malformed records,

1. Open the flow on the canvas.
2. Add a malformed records sink in your flow.
 - a) Create your job by defining your input file and source stage and adding services and subflows to your flow.
 - b) You can:
 - Connect a sink stage to the optional output port on the source stage in your flow. The optional port is the clear output port just beneath the black output port on your source stage. If you mouse over this port, you will see a tool tip that says, "error_port." Malformed records go to this sink.
 - Connect nothing to the optional output port on the source stage in your flow, ignoring all malformed records.
3. By default, processing stops at malformed records. This default behavior can be changed in your Advanced configuration options or in Spectrum Management Console. Regardless of your system's default behavior, you can override the default behavior for a job by following these steps:
 - a) Open the job in Spectrum Flow Designer.
 - b) Within an open job, go to **Edit > Job Options**.
 - c) Select either **Do not terminate the job on a malformed record** or select **Terminate the job after encountering this many malformed records** and enter the number of malformed records you will allow a job to encounter before terminating.

Defining Service Input

The Input stage defines the input fields for a service or subflow. It also defines test data to use during data inspection.

Input Fields Tab

This tab lists the fields that the dataflow accepts as input. If the Input stage is connected to another stage in the dataflow, a list of fields used by the stages in the dataflow is shown. For more information, see [Defining Input Fields for a Service or Subflow](#) on page 22.

Inspection Input Tab

This tab allows you to specify test input records to use with the Data Inspection tool. For more information about data inspection, see [Inspecting a flow](#) on page 815.

Defining Input Fields for a Service or Subflow

To define the input fields for a service or subflow, use the Input stage.

Note: If you define hierarchical data in the input fields, you cannot import data or view the data vertically.

1. Drag an Input stage to the canvas.
2. Connect the Input stage to the next stage in the dataflow.
3. Double-click the Input stage.
4. Select the fields you want to use for input. The list of fields shown depends on the stage that the Input stage is connected to.
5. To add a new field to the field list, click **Add**. The **Add Custom Field** window appears.
6. Click **Add** again.
7. In the **Field name** field, enter the name you want to use for this field.
8. Select the data type.

These data types are supported:

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.

bytearray	An array (list) of bytes. Note: Bytearray is not supported as an input for a REST service.
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 $(2-2^{-52}) \times 2^{1023}$. 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 $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -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).
list	Strictly speaking, a list is not a data type. However, when a field contains hierarchical data, it is treated as a "list" field. In Spectrum Technology Platform a list is a collection of data consisting of multiple values. For example, a field Names may contain a list of name values. This may be represented in an XML structure as: <pre><Names> <Name>John Smith</Name> <Name>Ann Fowler</Name> </Names></pre> <p>It is important to note that the Spectrum Technology Platform list data type is different from the XML schema list data type in that the XML list data type is a simple data type consisting of multiple values, whereas the Spectrum Technology Platform list data type is similar to an XML complex data type.</p>
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.

You can also add a new, user-defined data type if necessary, and that new type can be a list of any defined data type. For example, you could define a list of names (string), or a new data type of addresses that includes AddressLine1 (string), City (string), StateProvince (string) and PostalCode (string). After you create the field, you can view the data type by accessing the Input

Options dialog and pressing the button in the Data Type column. The **Data Type Details** dialog box will appear, showing the structure of the field.

9. Press **OK** again.
10. Click the **Expose** column check box to make the field available for stage operations. Clearing the check box and clicking **OK** deletes the field from the field list.
11. The **Data type name** field displays the default element name to use for input records in SOAP and REST web service requests to this service. The default is `Row`. If you want to use a different element name for input records, enter it here.

For example, with the default value `Row`, a JSON web service request would use `Row` as the element name for the input record, as shown here:

```
{
  "Input":
  {
    "Row": [
      {
        "AddressLine1": "1825 Kramer Ln",
        "City": "Austin",
        "StateProvince": "TX"
      }
    ]
  }
}
```

If you were to change the value in the **Data type name** field to `Address`, the JSON request would need to use `Address` instead of `Row` as the element name for the record, as shown here:

```
{
  "Input":
  {
    "Address": [
      {
        "AddressLine1": "1825 Kramer Ln",
        "City": "Austin",
        "StateProvince": "TX"
      }
    ]
  }
}
```

Defining A Web Service Data Type

The **Data type name** field allows you to control the WSDL (SOAP) and WADL (REST) interfaces for the service you are creating. The name of the Rows element is determined by the name you give this stage in the service, and the name of the Row element is determined by the text you enter here.

Note: For WSDL, both requests and responses are affected, but for WADL only responses are affected.

Prior to naming this stage and entering text in this field, your code might look like this:

```
<Rows>
  <Row>
    <FirstName>John</FirstName>
    <LastName>Doe</LastName>
  </Row>
  <Row>
    <FirstName>Jane</FirstName>
    <LastName>Doe</LastName>
  </Row>
</Rows>
```

After naming this stage and entering text in this field, your code might look like this:

```
<Names>
  <Name>
    <FirstName>John</FirstName>
    <LastName>Doe</LastName>
  </Name>
  <Name>
    <FirstName>Jane</FirstName>
    <LastName>Doe</LastName>
  </Name>
</Names>
```

Fields

Flat and Hierarchical Data

Spectrum Technology Platform supports flat data and hierarchical data. In general you can use either flat or hierarchical data as input and output for a flow. A few stages in Spectrum Spatial Routing require data to be in a hierarchical format.

Flat Data

Flat data consists of records, one on each line, and fields in each record. Fields are delimited by a specific character or positioned in a defined location on the line. For example, this is flat data with comma-delimited fields:

```
Sam,43,United States
Jeff,32,Canada
Mary,61,Ireland
```

To read flat data into a flow, you can use the Read from File, Read from DB, or Input stages. To write flat data output from a flow, you can use the Write to File, Write to DB, or Output stages.

Hierarchical Data

Hierarchical data is a tree-like structure with data elements that have parent-child relationships. Spectrum Technology Platform can read and write hierarchical data in XML and Variable Format File format. For example, this shows hierarchical data in XML:

```
<customers>
  <customer>
    <name>Sam</name>
    <age>43</age>
    <country>United States</country>
  </customer>
  <customer>
    <name>Jeff</name>
    <age>32</age>
    <country>Canada</country>
  </customer>
  <customer>
    <name>Mary</name>
    <age>61</age>
    <country>Ireland</country>
  </customer>
</customers>
```

This example shows a structure where `<customer>` represents a record and each record consists of simple XML elements (`<name>`, `<age>`, and `<country>`).

Converting Data

There are many cases where you might need to convert data from flat to hierarchical, or from hierarchical to flat. For example, you may have data flow input in hierarchical format but want the data flow to output flat data. You may also need to convert flat input data to hierarchical data for certain stages (especially stages in Spectrum Spatial) then convert the data back to flat data for output.

To convert data from flat to hierarchical you can use:

- The Process List tool
- The Aggregator stage in a flow

To convert data from hierarchical to flat use the Splitter stage.

Converting flat data to a list

Process List is a tool you can use within a service or subflow to turn flat data into a list.

This feature is useful if your dataflows include stages that require list input, such as those in Spectrum Spatial.

1. With an existing flow in place, right-click the stage whose output you want to convert into a list. This could be any stage except Input or Output.
2. Select **Process List**. You will see the stage within a blue square background.
3. To move a stage into and out of the process list, press the **Shift** key while dragging the additional stage.

Note: If you have several stages whose data you would like Process List to handle, consider creating a subflow, bringing it into your flow, and applying the Process List feature to the subflow as a whole.

4. The input and output fields of a process list are called "ListField." Using the Rename Fields function, you must map your input stage field to "ListField" in the input channel, and map "ListField" to your output stage field. For more information, see [Changing a field name](#) on page 39.
5. If you want the list to keep the data in original input order, right-click the Process List box and select **Options**.
6. Check the **Maintain sort order** box.
7. To confirm that the data input into the next stage will be formatted as a list, validate or inspect the flow. For more information about inspecting data, see [Inspecting a flow](#).

Data Types

Spectrum Technology Platform supports a variety of numeric, string, and complex data types. Depending on the type of processing you want to perform you may use one or more of these. For an address validation flow you might only use string data. For flows that involve the mathematical computations you may use numeric or Boolean data types. For flows that perform spatial processing you may use a complex data type. For flows that combine these, you may use a variety of data types.

Specifying a Field's Data Type

You can specify the data type for a field in these situations:

- **Source stages:** Specifying data types allows you to set the data type at the beginning of a flow, eliminating the need for data type conversions later in the flow. Note that for Read from DB, the data type is selected automatically and cannot be changed.
- **Sink stages:** Specifying data types allows you to control the data format returned by the flow. Note that for Write to DB, the data type is selected automatically and cannot be changed.
- **Transformer stage:** You can specify data types in this stage if you use a custom script.
- **Math stage and Group Statistics stage:** Since these stages perform mathematical calculations, choosing to use a particular numeric data type can have an effect on the results of the calculations, such as the precision of a division operation. If you specify a data type for a field that is different than the data type of the field coming into the stage, the downstream channel will automatically convert the field to the data type you specify, as described in [Automatic Data Type Conversion](#) on page 28.

Note: Each stage supports different data types. For a description of the supported data types for each stage, see the documentation for a specific stage.

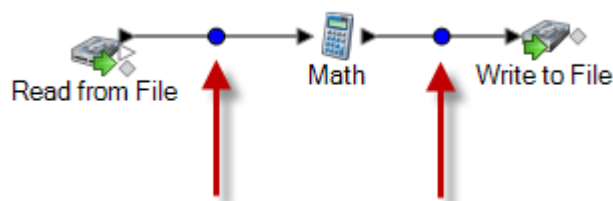
Related reference

Data types

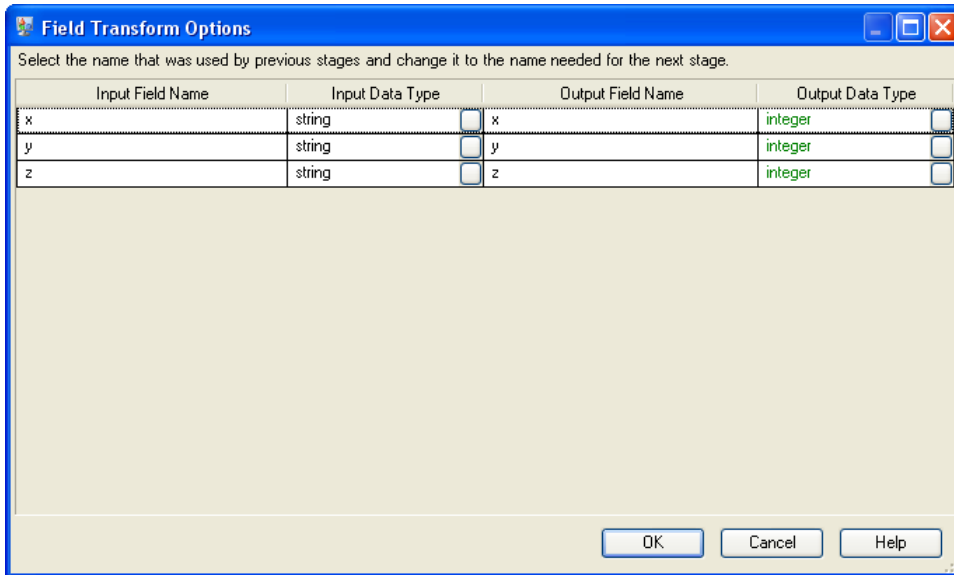
Automatic Data Type Conversion

When the data presented to a stage is of an inappropriate type, Spectrum Technology Platform can, in some cases, automatically convert the data to the appropriate type. For example, Validate Address accepts only string data as input. If the PostalCode input field is of type integer, Spectrum Technology Platform can automatically convert the field to string and successfully process the PostalCode field. Likewise, the Math stage needs data to be of a numeric data type. If the incoming data is of type string, Spectrum Technology Platform can convert the data to the data type specified in the Math stage's Fields tab.

Automatic data type conversions happen in the channels of a flow. If a channel is successfully converting a data type, there will be a blue dot in the middle of the channel:

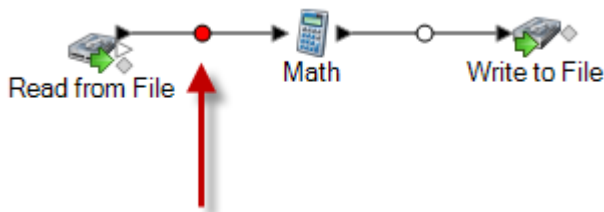


If you double-click the channel you can see the data type conversion that's occurring. In this case, string data is being converted to integer data:



Note that you cannot change the data type in this dialog box for automatic data type conversions. The output data type is determined by settings in the downstream stage.

Fields that do not contain valid values or that cannot be converted result in a red circle in the channel.



You can specify what the flow should do if type conversion fails by using the type conversion options.

Setting Data Type Conversion Options for a Flow

Data type conversion occurs when a flow automatically converts a field to the data type needed by a stage. Data type conversion also occurs when within some stages. For example, in Read from DB you can choose to have a field use the string data type even though the source data is in a numeric data type. The data is converted into the string data type when it is read into the flow.

There are two settings that you can use to control data type conversions. First, there are settings that determine how to format numeric, date, and time data converted into a string. For example, you may want date data that is converted into a string to be represented in the format mm/dd/yyyy rather than dd/mm/yyyy. The other setting controls what should happen if the system is unable to convert a field from one data type to another.

The default data type conversion settings for your system are specified in Management Console. You can override the default formats for individual flows in Spectrum Enterprise Designer.

This procedure describes how to override the default data type conversion options for a flow.

Note: Subflows inherit the type conversion settings from the flow they are in. You cannot specify type conversion settings for subflows.

1. Open the flow in Spectrum Enterprise Designer.
2. Select **Edit > Type Conversion Options**.
3. Check the box **Override system default options with the following values**.
4. 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.

5. 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 32. To specify your own number format, type the format into the file using the notation described in [Number Patterns](#) on page 34.

6. Under **Null handling**, choose what to do if a field that needs type conversion contains a null value. If you select any of the options below, either the flow or the record containing the null value will fail based on whether you selected **Fail the flow** or **Fail the record** under **Type Conversion Failures**.

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.

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
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

Letter	Description	Example
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
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

Letter	Description	Example
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, 1.234×10^3 .

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 characters as literal characters in your number pattern's prefix or suffix, surround the special character with quotes.

Symbol	Description
0	<p>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:</p> <p>0000</p> <p>Would be:</p> <p>0027</p>
#	<p>Represents a digit but zeros are omitted. For example, the number twenty-seven when applied to this pattern:</p> <p>####</p> <p>Would be:</p> <p>27</p>
.	<p>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.</p>
-	<p>The negative sign used in the selected locale. For most locals this is the minus sign (-).</p>
,	<p>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.</p> <p>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:</p> <p>#, ##, ###, ####</p> <p>#####, ####</p> <p>##, #####, ####</p>
E	<p>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 35.</p>
;	<p>Separates positive and negative subpatterns. See Patterns for Negative Numbers on page 35.</p>

Symbol	Description
%	<p>Multiply the number by 100 and show the number as a percentage. For example, the number .35 when applied to this pattern:</p> <pre>##%</pre> <p>Would produce this result:</p> <pre>35%</pre>
¤	<p>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.</p>
'	<p>Used to quote special characters in a prefix or suffix. For example,</p> <pre>''##''</pre> <p>Formats 123 to:</p> <pre>"#123"</pre> <p>To create a single quote itself, use two in a row:</p> <pre>"# o''clock"</pre>

**Changing a field's data type*

Spectrum Technology Platform automatically changes field data types as needed using the type conversion settings specified in Spectrum Management Console, or the dataflow type conversion options specified in Spectrum Enterprise Designer. In most situations you do not need to manually change field data types because any necessary data type conversions are handled automatically. However, in cases where a stage is unable to convert incoming data to the necessary data type, you may need to manually change the data type in the upstream channel.

There are only a few possible type conversions that you can perform manually. Those are:

- Polygon and MultiPolygon types can be converted to and from a geometry type.
- Date, time, and datetime data types can be converted to and from a string type.

To manually change a field's data type, follow this procedure.

1. In Spectrum Enterprise Designer, double-click the channel where you want to change the field's data type. A channel is the line that connects two stages on the canvas.
2. Click the small square button next to the data type that you want to change.

Note: If a small square button is not visible next to the data type, then manual data type conversion is not available for your situation.

3. For date, time, and datetime data types:

Note: Only the appropriate options will be displayed depending on the data type chosen.

- 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`.

4. Click **OK**.

The color of the data type name changes to green.

5. Click **OK** again to save the change.

Changing a field name

There are a variety of situations where you may need to rename a field in a flow. For example:

- A stage's input requires certain field names but the previous stage's output uses other field names.
- There is data in a field which you want to preserve when a downstream stage writes data to a field of the same name.

Note: After a field is renamed, it is no longer available in subsequent stages with the old name.

1. In a flow, double-click the channel between two stages. The **Field Transform Options** dialog box appears.
2. Change the field name or names as desired.

For example, the latter stage could require "AddressLine3" but the former stage uses "FirmName" instead. In this case, you would click the drop-down arrow in the Input Field Name that corresponds to AddressLine3 as the Output Field Name and select "FirmName."

The color of the output field name changes to green.

3. Click **OK**.

Reserved Field Names

Flow designer reserves these field names, so do not use these names in your flows:

- Status
- Status.Code
- Status.Description

Control Stages

Use control stages to move data along different paths in a flow, to split or group records, and to perform basic data transforms and mathematical operations.

Aggregator

Aggregator converts flat data to hierarchical data. It takes input data from a single source, creates a schema (a structured hierarchy of data) by grouping the data based on fields you specify, then constructs the groups in the schema.

Note: You cannot configure this stage in the technical preview version of Spectrum Flow Designer.

Aggregator converts flat data to hierarchical data. It takes input data from a single source, creates a schema (a structured hierarchy of data) by grouping the data based on fields you specify, then constructs the groups in the schema.

Note: If your data includes a field by which you will group your data, such as an ID field, you must sort your data before running it through an Aggregator. You can do this by sorting the data prior to bringing it into the flow, by sorting the input file within Spectrum Enterprise Designer (for jobs or subflows, but not services) or by adding a Sorter stage to your flow (for jobs, services, or subflows).

Group By

Choose the field you want to use as the basis for aggregating into a hierarchy by selecting **Group by** in the tree then clicking **Add**. Records that have the same value in the field you choose will have their data aggregated into a single hierarchy. If you select multiple fields then the data from all fields must match in order for the records to be grouped into a hierarchy.

For example, if you want to group data by account number you would select the account number field. All incoming records that have the same value in the account number field would have their data grouped into a single hierarchical record.

Note: You must connect a stage to the Aggregator input port in order for a list of fields to be available to choose from.

Output Lists

The fields you choose under **Output lists** determine which fields are included in each record created by Aggregator. To add a field, select **Output lists** then click **Add** and choose one of these options:

- | | |
|-----------------------|---|
| Existing field | Select this option if you want to add a field from the flow to the hierarchy. |
| New data type | Select this option if you want to create a parent field to which you can then add child fields. |
| Template | This option allows you to add a field based on data in the stage connected to the Aggregator's output port. |

If you want the field to have child fields, check the **List** box.

Enter the name of the field in the **Name** text box, or leave it as-is if it auto-filled and you are satisfied with the name. Keep in mind that the Aggregator stage does not allow invalid XML characters in field names; it does allow alphanumeric characters, periods (.), underscores (_), and hyphens (-).

Click **Add** to add the field. You can specify another field to add to the same level in the hierarchy or you can click **Close**.

To add child fields to an existing field, select the parent field then click **Add**.

Note: You can modify the field group by highlighting a row and clicking **Modify**, and you can remove a field group by highlighting a row and clicking **Remove**. You can also change the order of fields by clicking a field and clicking **Move Up** or **Move Down**.

Broadcaster

A Broadcaster takes a stream of records and splits it into multiple streams, allowing you to send records to multiple stages for simultaneous processing.

Broadcaster has no settings to change.

Conditional Router

The **Conditional Router** stage sends records to different paths in the flow depending on the criteria you specify. The stage can have one or more output ports, depending on the defined criteria. Output ports are numbered consecutively, starting with 1 (which displays as "port").

The output ports connect to different stages to which the data is to be sent, depending on defined conditions. For example, you can send one set of records to port 1 in case of a successful match, while a different set of records can be sent to port 2 in case of a failed match.

An input record is written to the Conditional Router's output port only if the entire expression evaluates to true.

Configuring a Conditional Router

1. Under **Control Stages**, click on **Conditional Router** and drag it to the canvas, placing it in the desired location within the flow.
2. Connect the router to other stages on the canvas.

Note: This is a mandatory step before defining the port settings. Otherwise the ports are not available for editing.

3. Double-click on the **Conditional Router** stage on the canvas. The **Conditional Router Options** window appears.

4. Click the square button in the **Condition/Expression** column against the **port** row. The **Expressions Editor** window appears.
5. In the **Choose Expression Type** section, select one of the following:
 - **Expression created with Expression Builder:** Select this option to create a basic expression, where you can add Groups and Expressions, which can be combined using different logical operators. For more information, see [Using the Expression Builder](#) on page 42.
 - **Custom expression:** Select this option to write an expression using the Groovy scripting language. For more information, see [Writing a Custom Expression](#) on page 45.
 - **Default expression:** Select this to route records to this port by default. Records that do not match any of the other ports' expressions will be routed to this port. You should always have an output port with "default" as the expression to ensure that no rows are missed in case of a port mismatch, and all rows are written from the router.
6. Click **OK**. The **Expressions Editor** window closes.
7. Click **OK** on the **Conditional Router Options** window.

Using the Expression Builder

The **Expression Builder** of the **Conditional Router** stage allows you to create an expression that must evaluate to true for an input record to be routed to the output port of the stage.

1. Each parent group comprises of a desired conditional combination of child expressions and child groups.
2. Each expression consists of a left operand, a right operand and a logical operator.
3. Each group must specify whether *all* or *any* of its constituent conditions must hold true for the entire group to evaluate to true.

To build an expression using the Expression Builder:

1. In the **Expression Editor**, select the option **Expression created with Expression Builder**. By default, the Expression Builder option is selected and a parent group is displayed in the expression hierarchy tree on the left of the **Expression Builder** section.
2. To add a child group within the selected group, click **Add Group** . This newly added group gets added as a child of the parent group, and is selected in the tree by default. Within each group, you can add child expressions and child groups.
3. For each group, select either **All true** or **Any true** under the **Combine expression method** header.
 - **All true:** The group evaluates to true only if all the child criteria of the group hold true.
 - **Any true:** The group evaluates to true if even one of its child criteria hold true.
4. To add a child expression within the selected group, click **Add Expression**. The newly added expression gets added as a child of the parent group and is selected in the tree by default.

To define this child expression:

- a) Specify the left operand of the selected expression using the **Field** dropdown to select any one of the columns in the input file.
- b) Specify the logical operator connecting the two components of the selected expression by selecting the appropriate operator from the **Operator** field as explained below:

Table 1: Expression Builder Operators

Operator	Description
Is Equal	Checks if the value in the field matches the value or field specified.
Is Not Equal	Checks if the value in the field does not match the value or field specified.
Is Null	Checks if the field is a null value.
Is Not Null	Checks if the field is not a null value.
Is Empty	Checks if the field is null or a string with a length of 0. Note: This operation is only available for fields with a data type of string.
Is Not Empty	Checks if the field is neither null nor a string with a length of 0. Note: This operation is only available for fields with a data type of string.
Is Less Than	Checks if the field has a numeric value that is less than the value specified. This operator works on numeric data types as well as string fields that contain numbers. Note: This operation is not available for fields with a data type of Boolean.
Is Less Than Or Equal To	Checks if the field has a numeric value that is less than or equal to the value specified. This operator works on numeric data types as well as string fields that contain numbers. Note: This operation is not available for fields with a data type of Boolean.

Operator	Description
Is Greater Than	<p>Checks if the field has a numeric value that is greater than the value specified. This operator works on numeric data types as well as string fields that contain numbers.</p> <p>Note: This operation is not available for fields with a data type of Boolean.</p>
Is Greater Than Or Equal To	<p>Checks if the field has a numeric value that is greater than or equal to the value specified. This operator works on numeric data types as well as string fields that contain numbers.</p> <p>Note: This operation is not available for fields with a data type of Boolean.</p>
Starts With	<p>Checks if the field begins with the characters specified.</p> <p>Note: This operation is only available for fields with a data type of string.</p>
Does Not Start With	<p>Checks if the field does not begin with the characters specified.</p> <p>Note: This operation is only available for fields with a data type of string.</p>
Contains	<p>Checks if the field contains the string specified.</p> <p>Note: This operation is only available for fields with a data type of string.</p>
Does Not Contain	<p>Checks if the field does not contain the string specified.</p> <p>Note: This operation is only available for fields with a data type of string.</p>
Ends With	<p>Checks if the field ends with the characters specified.</p> <p>Note: This operation is only available for fields with a data type of string.</p>
Does Not End With	<p>Checks if the field ends with the characters specified.</p> <p>Note: This operation is only available for fields with a data type of string.</p>

Operator	Description
Matches Regular Expression	Matches the field with a regular expression for identifying strings of text of interest, such as particular characters, words, or patterns of characters. The value field should contain a valid regular expression pattern. Note: This operation is only available for fields with a data type of string.

- c) Specify the right operand of the selected expression by selecting either `Value` or `Field`.
- `Value`: The left operand of the selected expression is compared to this value.
 - `Field`: The left operand of the selected expression is compared to this column of the same input file. Select the right operand column from the dropdown.
5. To add a sibling expression or sibling group to any entity, select that entity in the tree and click **Add Expression** or **Add Group** respectively.
 6. To shift a child expression or child group from one parent group to a different parent group, drag it to the desired parent group header in the criteria tree on the left.
 7. Repeat the above steps to add as many child expressions and child groups as are required to create the desired final expression criteria.
 8. Click **OK**.

The **Condition/Expression** column in the **Conditional Router Options** window displays the defined expression criteria, which must evaluate to true for a record to be written to the stage's corresponding output port.

Writing a Custom Expression

You can write your own custom expressions to control how Conditional Router routes records using the Groovy scripting language to create an expression.

Using Groovy Scripting

For information about Groovy, see groovy-lang.org.

Groovy expressions used in the Conditional Router stage must evaluate to a Boolean value (true or false) which indicates whether the record should be written to the port. The record is routed to the first output port whose expression evaluates to true.

For example, if you need to route records with a validation confidence level of ≥ 85 to one stage and records with a validation confidence level of < 85 to another stage, your script would look like:

```
data['Confidence'] >= 85
```

The script for the other port would look like:

```
data['Confidence'] < 85
```

The router would evaluate the value of the Confidence field against your criteria to determine which output port to send it to.

Checking a Field for a Single Value

This example evaluates to true if the Status field has 'F' in it. This would have to be an exact match, so 'f' would not evaluate to true.

```
return data['Status'] == 'F';
```

Checking a Field for Multiple Values

This example evaluates to true if the Status field has 'F' or 'f' in it.

```
boolean returnValue = false;
if (data['Status'] == 'F' || data['Status'] == 'f')
{
    returnValue = true;
}
return returnValue;
```

Evaluating Field Length

This example evaluates to true if the PostalCode field has more than 5 characters.

```
return data['PostalCode'].length() > 5;
```

Checking for a Character Within a Field Value

This example evaluates to true if the PostalCode field has a dash in it.

```
boolean returnValue = false;
if (data['PostalCode'].indexOf('-') != -1)
{
    returnValue = true;
}
return returnValue;
```

Scripting Guidelines

1. Column names must be enclosed within either single or double quotes.

For example, this syntax is incorrect because the column name `PostalCode` is not enclosed within either single or double quotes.

```
return data[PostalCode];
```

2. A column name must be specified.

For example, this syntax is incorrect because no column is specified.

```
return data[];
```

3. A `return` statement must return a `Boolean` value.

For example, this script is incorrect because `row.set('PostalCode', '88989')` does not return a `Boolean` value. It just sets the value of the `PostalCode` field to `88989`.

```
return row.set('PostalCode', '88989');
```

4. Use a single equals sign (`=`) to set the value of a field, and a double equals sign (`==`) to check the value of a field.

Group Statistics

The Group Statistics stage allows you to run statistical operations across multiple data rows broken down into groups that you want to analyze. If no groups are defined all rows will be treated as belonging to one group.

Groups are defined by one or more fields that have the same value across multiple data rows.

For example, the data in this table could be grouped by region, state, or both.

Region	State
East	MD
East	MD
East	CT
West	CA
West	CA

A group by Region would yield East and West. A group by State would yield California, Connecticut, and Maryland. A group by Region and State would yield East/Maryland, East/Connecticut, and West/California.

Input

The Group Statistics stage takes any field as input. Grouping can be performed on numeric or string data.

Options

Table 2: Operations Tab

Option	Description
Input fields	Lists the fields in the flow that you can use to group records and perform calculations.
Row	<p>Specifies the field or fields you want to use as categories for the calculations. For example, if you had data that included a Region field and you wanted to calculate total population by region, you would group by the Region field.</p> <p>To add a field, select the field in the Input fields list then click >>.</p>
Column	<p>Optional. For creating a pivot table, specifies the field or fields whose values you want to pivot into columns for the purposes of cross tabulation.</p> <p>To add a field, select the field in the Input fields list then click >>.</p> <p>For example, if you had data that includes regions and shipping dates, and you want to tally the number of shipments each day for each state, you must specify the state field as a row and the shipment date field as a column.</p>
Rows and Columns are presorted in the configured order	<p>Indicates that the input data is already sorted.</p> <p>If this check box is checked, the stage does not sort the data and performs the specified operation directly on the input data.</p>
Operation	<p>Specifies the calculation to perform on each group. To add an operation, select the field in the Input fields list that you want to use for the operation then click >>.</p> <p>For more information about the supported Group Statistics operations, see Operations on page 50.</p>

Option	Description
Type	For the input and output fields, specifies the data type. <ul style="list-style-type: none"> 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) Float A numeric data type that contains both negative and positive single precision numbers between 2^{-149} (1.4E-45) and $(2-2^{-23})\times 2^{127}$ (3.4028235E38) Double A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} (4.9E-324) and $(2-2^{-52})\times 2^{1023}$ (1.7976931348623157E308) <p>Note: When using the integer and long types, data can be lost if the input number or calculated number from an operation contains decimal data.</p>
Get count of records that are computed upon	Returns the actual number of records in a group on which the selected operation is performed. <p>This column <i>Computational Count</i> excludes those input records where the column on which the operation is performed contains <code>null</code> values.</p>

Fields Tab

The Fields tab is used when creating a pivot table. For more information, see [Creating a Pivot Table](#) on page 55.

Output Tab

Option	Description
Return one row for each group	For each group of rows, return a single row that contains the aggregated data for all rows in the group. Individual rows will be dropped. If this option is not selected, all rows will be returned. No data will be dropped. <p>This option is not available if you use the Percent Rank or ZScore operations.</p>
Return a count of rows in each group	Returns the number of rows in each group. The default output field name that will contain the count is GroupCount.

Option	Description
Return a unique ID for each group	Returns a unique ID for each group of rows. The ID starts at 1 and increments by 1 for each additional group found. The default field name is GroupID.

Operations

The calculations available are:

Average	For each group, calculates the average value of a given field. For example, if you had a group of records with values 10, 12, 1, and 600 in a given field, the average value of that field for that group would be 155.75, calculated as $(10+12+1+600)\div 4$.
Maximum	For each group, returns the largest value in a given field. For example, if you had a group of records with values 10, 12, 1, and 600 in a given field, the maximum value of that field for that group would be 600.
Minimum	For each group, returns the smallest value in a given field. For example, if you had a group of records with values 10, 12, 1, and 600 in a given field, the minimum value of that field for that group would be 1.
Percent Rank	For each record within a group, calculates the percentile rank of a value in a given field relative to other records in the group. The percentile rank represents the percentage of records in the group with lower values in the field.
Percentile	For each group, calculates the value that would represent the percentile you specify (0 - 100) for a given field. A percentile represents the percentage of records that have a lower score. For example, if you have a group of records with values 22, 26, and 74, and you perform a percentile calculation specifying the 60th percentile, the operation would return 35.6. This means that a record with a value of 35.6 in the given field would be in the 60th percentile of records in the group.
Standard Deviation	For each group, calculates the standard deviation for a given field. The standard deviation measures the amount of dispersion within the group. The lower the standard deviation, the more the values are centered around the mean value, and therefore the less dispersed the values. The higher the value, the more widely dispersed the values. The standard deviation is expressed in the same units as the data. The standard deviation is the square root of the variance.
Sum	For each group, calculates the sum of the values for a given field.
Variance	For each group, calculates the variance for a given field. The variance measures the amount of dispersion within the group. It is the square of the standard deviation.
ZScore	For each record in a group, returns the ZScore. The ZScore indicates how many standard deviations a value is above or below the group's mean.
Alphabetical First	For each group, returns first dictionary value. If there are more than one field values having same length or dictionary position, it returns the first occurrence of

that value. For example, if group has a record with values Joel and Joey in a field, then the alphabetical first value for a group will be Joel as l comes before y in alphabet.

Alphabetical Last	For each group, returns last dictionary value. If there are more than one field values having same length or dictionary position, it returns the last occurrence of that value. For example, if group has a record with values Joel and Joey in a field, then the alphabetical last value for a group will be Joey as y comes after l in alphabet.
Longest	For each group, returns longest value. For example, if group has a record with values Joel and Jacob in a field, then the longest length value for a group will be Jacob as it has 5 alphabets whereas Joel has 4.
Shortest	For each group, returns shortest value. For example, if group has a record with values Joel and Jacob in a field, then the shortest length value for a group will be Joel as it has 4 alphabets whereas Jacob has 5.
Latest	For each group, returns the latest date or datetime value. For example, if a group has a record with values 15-12-2014 and 24-12-2014 in a field, then the latest value for the group is 24-12-2014.
Earliest	For each group, returns the earliest date or datetime value. For example, if a group has a record with values 15-12-2014 and 24-12-2014 in a field, then the earliest value for the group will be 15-12-2014.

Output Columns

Field Name	Description and Valid Values
<i>OperationOfInputFieldName</i>	Contains the result of a calculation. Group Statistics creates one output field for each operation and names the field based on the operation and field. For example, the default field name for a Sum operation performed on a field named <code>Population</code> would be <code>SumOfPopulation</code> .
<i>Value_Operation</i>	Contains the result of a pivot, where <i>Value</i> is one of the values in a pivot column and <i>Operation</i> is the operation performed on the column. For more information, see Creating a Pivot Table on page 55.
<code>GroupCount</code>	Indicates the number of records in the group.
<code>GroupID</code>	A unique number assigned to each group sequentially. The first group has a <code>GroupID</code> value of 1, the second has a value of 2, and increments accordingly.

Field Name	Description and Valid Values				
<code>ComputationalCountOperationOfInputFieldName</code>	Indicates the actual number of records in a group on which the operation is performed. For example, for the operation <code>Average</code> performed on the <code>Salary</code> column, the column <code>ComputationalCountAverageOfSalary</code> is generated.				
<code>Status</code>	Reports the success or failure of the Group Statistics calculations. <table border="0"> <tr> <td>null</td> <td>Success</td> </tr> <tr> <td>F</td> <td>Failure</td> </tr> </table>	null	Success	F	Failure
null	Success				
F	Failure				
<code>Status.Code</code>	Reason for the processing failure. The status codes available are: <table border="0"> <tr> <td>UnableToDoGroupStatistics</td> <td>The Group Statistics stage was unable to perform its calculations.</td> </tr> <tr> <td>Error calculating percentile value</td> <td>The percentile value could not be calculated using the input data provided.</td> </tr> </table>	UnableToDoGroupStatistics	The Group Statistics stage was unable to perform its calculations.	Error calculating percentile value	The percentile value could not be calculated using the input data provided.
UnableToDoGroupStatistics	The Group Statistics stage was unable to perform its calculations.				
Error calculating percentile value	The percentile value could not be calculated using the input data provided.				
<code>Status.Description</code>	A verbose description of the error. <p>The input field value could not be converted to the field type. It might be overflow!</p> <p>A number in an input field is larger than the data type allows. Try converting to a data type that supports larger numbers, such as double.</p>				

Group Statistics Example

This input data shows the number of customers you have in certain counties. The data also shows the U.S. state in which the county is located (MD, VA, CA, and NV), as well as the region (East or West). The first row is a header record.

```
Region|State|County|Customers
East|MD|Calvert|25
East|MD|Calvert|30
East|MD|Prince Georges|30
East|MD|Montgomery|20
East|MD|Baltimore|25
East|VA|Fairfax|45
East|VA|Clarke|35
West|CA|Alameda|74
West|CA|Los Angeles|26
West|NV|Washoe|22
```

If you wanted to calculate the total number of customers for each region, you would define the `Region` field as a row in the **Operations** tab. For the operation, you would perform a sum operation on the `Customers` field.

The result:

```
Region|SumOfCustomers
East|210.0
West|122.0
```

Note: This example shows a basic group statistics operation using only rows to aggregate data. You can also create a pivot table, which aggregates both rows and columns, by specifying a column to group by in the **Operations** tab. For more information about creating a pivot table, see [Creating a Pivot Table](#) on page 55.

Pivot Tables

A pivot table aggregates and transposes column values in the flow to make it easier to analyze data visually. With pivot, you can arrange input columns into a cross tabulation format (also known as crosstab) that produces rows, columns and summarized values. You can also use fields as input and not display them. You can use pivot to pivot on two dimensions or to group aggregate data on one dimension.

This example shows sales data for shirts.

Table 3: Input Data

Region	Gender	Style	Ship Date	Units	Price	Cost
East	Boy	Tee	1/31/2010	12	11.04	10.42
East	Boy	Golf	6/31/2010	12	13.00	10.60
East	Boy	Fancy	2/25/2010	12	11.96	11.74
East	Girl	Tee	1/31/2010	10	11.27	10.56
East	Girl	Golf	6/31/2010	10	12.12	11.95
East	Girl	Fancy	1/31/2010	10	13.74	13.33
West	Boy	Tee	1/31/2010	11	11.44	10.94
West	Boy	Golf	2/25/2010	11	12.63	11.73

Region	Gender	Style	Ship Date	Units	Price	Cost
West	Boy	Fancy	2/25/2010	11	12.06	10.51
West	Girl	Tee	2/25/2010	15	13.42	13.29
West	Girl	Golf	6/31/2010	15	11.48	10.67
North	Boy	Tee	2/25/2010	17	16.04	10.42
North	Boy	Fancy	2/25/2010	12	11.56	12.42
North	Girl	Tee	2/25/2010	16	12.32	18.42
North	Boy	Golf	1/31/2010	18	11.78	13.23
North	Girl	Tee	2/25/2010	12	18.45	11.64
North	Girl	Golf	2/25/2010	14	11.23	19.85
North	Boy	Fancy	1/31/2010	16	12.54	13.42
North	Girl	Tee	2/25/2010	17	181.73	15.83
South	Boy	Fancy	1/31/2010	19	14.15	13.42
South	Girl	Tee	2/25/2010	11	11.85	12.92
South	Girl	Fancy	1/31/2010	13	11.54	14.35
South	Boy	Tee	2/25/2010	15	14.14	14.73
South	Boy	Golf	2/25/2010	16	17.83	17.83
South	Girl	Fancy	6/31/2010	11	18.24	12.35
South	Girl	Tee	1/31/2010	20	19.94	12.95
South	Boy	Golf	2/25/2010	12	21.25	19.56

We want to be able to determine how many units we sold in each region for every ship date. To do this, we use pivot to generate this table:

Table 4: Pivot Table

Region	1/31/2020_ShipDate	2/25/2020_ShipDate	6/31/2020_ShipDate
East	32	12	22
North	34	88	
South	52	54	11
West	11	37	15

In this case, the column is Ship Date, the row is Region, and the data we would like to see is Units. The total number of units shipped is displayed here using a sum aggregation.

Creating a Pivot Table

A pivot table summarizes data for easier analysis by creating table row and column categories based on input data. For more information, see [Pivot Tables](#) on page 53.

In the **Group Statistics** stage options:

1. In the **Operations** tab, select a field from **Input Fields** which contains the data you want to use as the row labels in your pivot table. Then click the >> button next to the **Rows** field.
2. Select a field that contains the data you want to use as the columns in your pivot table then click the >> button next to the **Columns** field.

Tip: At this point, run inspection to see the results of your selections. This will help you visualize the results of the cross tabulations based on the columns and rows you have selected.

3. To skip sorting the input records, check **Rows and Columns are pre-sorted in the configured order**.

If this field is checked, the stage processes the input records without sorting them.

Note: Check this if the records are already sorted.

4. To define the operation to be performed, click the >> button next to the **Operations** field.
In the **Add Operation** window:
 - a) Select the **Operation** to be performed.

- b) In the **Input Field** section, select the **Name** and **Type** of the input field on which the operation must be performed.
- c) In the **Output Field** section, enter the **Name** and select the **Type** of the output field to be generated once the operation is performed.
- d) To fetch the actual count of input records on which the operation is performed as a separate output column, check **Get count of records that are computed upon**.

Records with `null` values are not included in the count

`ComputationalCount<Operation>Of<InputFieldName>`.

Functions on which Computational Count is supported:

- Average
- Variance
- ZScore
- Standard Deviation
- Percentile
- Percent Rank
- Sum

For any other operation, this check box remains disabled.

- 5. To define the output fields for each column in the pivot table, click the **Fields** tab of the stage options.

Tip: In order to define fields accurately, run an inspection flow once, before this step, to see the column names generated by your data.

- a) Click **Add** to display the **Add Field** window.
- b) In the **Add Field** window, the grid columns are based on the **Columns** fields you chose in the **Operations** tab. In these grid columns, enter those values that you see as the column headings on running an inspection flow.

The records in the column Data can also be populated in one go by using the Import feature. To import data from a CSV or TXT file:

1. Click **Import**
2. Browse the source file using **File name** field
3. Enter the **Field and Record Separator** values
4. Click OK

All the records in the file get populated in the Column Data table.

Note: The source file should not have any header row.

For example, if you selected an input field called `ShipDate` in **Columns** in the **Operations** tab, the grid in the **Add Field** window would have a column labeled `ShipDate`. In this grid column, enter the exact `ShipDate` values present in your flow input data, such as `2/25/2010`, `1/31/2010`.

- c) In the **Operation** field, select one or more operations for which output columns are generated for each entered column field value. Note that the operation you select only affects the field name and does not control the actual calculation.

To change the operations listed in the **Operation** field of the **Fields** tab, modify the **Operations** field values in the **Operations** tab.

Attention: The Computational Count operation option `ComputationalCountOperationOfInputFieldName` is listed only if the **Get count of records that are computed upon** check box is selected while defining the **Operation** in the **Operations** tab.

- d) Click **Add**.

6. Click **OK**.

For each input value you entered in the grid above, output columns are automatically created by mapping those against each of the selected **Operation** values. A *Cartesian product* of the entered input column values in the grid and the selected Operations is used to automatically generate the final output columns.

The names of these output columns follow the naming convention

`Data_OperationOfInputFieldName`, where *Data* is the value you specified in the first field, *Operation* is the operation you selected in the **Operation** field, and *InputFieldName* is input column on which the *Operation* is performed.

Pivot Table Example

The input data which shows shipping information from the fulfillment department:

```
Region,State,County,ShipDate,Unit
East,MD,Calvert,1/31/2010,
East,MD,Calvert,6/31/2010,212
East,MD,Calvert,1/31/2010,633
East,MD,Calvert,6/31/2010,234
East,MD,Prince Georges,2/25/2010,112
East,MD,Montgomery,1/31/2010,120
East,MD,Baltimore,6/31/2010,210
East,VA,Fairfax,1/31/2010,710
West,CA,SanJose,1/31/2010,191
West,CA,Alameda,2/25/2010,411
West,CA,Los Angeles,2/25/2010,
West,CA,Los Angeles,2/25/2010,215
West,CA,Los Angeles,6/31/2010,615
West,CA,Los Angeles,6/31/2010,727
```

To determine the number of shipments that went out on each shipping date for each state, create a pivot table by configuring the Group Statistics stage as:

- Operations tab > Input Fields = County, Region, ShipDate, State, Unit
 - Rows = State
 - Columns = ShipDate

- Operations = Assign Sum of Unit to SumOfUnit
- Fields tab > Stage options = Add the exact dates in the grid that appear in the ShipDate field of the flow input data, and select the **Operation** values to be displayed for each of the column values.
- Fields tab >

On clicking **OK** in the **Add Field** window, the output columns to be created are automatically listed in the **Fields** tab. These output columns are a *Cartesian product* of the exact input values and the operations you selected in the **Add Field** window.

Output

```
State,1/31/2010_GroupCount,1/31/2010_ComputationalCountSumOfUnit,
1/31/2010_SumOfUnit,2/25/2010_GroupCount,2/25/2010_ComputationalCountSumOfUnit,
2/25/2010_SumOfUnit,6/31/2010_GroupCount,6/31/2010_ComputationalCountSumOfUnit,
6/31/2010_SumOfUnit
VA,1,1,710,,,,,
CA,1,1,191,3,2,626,2,2,1342
MD,3,2,753,1,1,112,3,3,656
```

Math

The Math stage handles mathematical calculations on a single data row and allows you to conduct a variety of math functions using one or more expressions. Data is input as strings but the values must be numeric or Boolean, based on the type of operation being performed on the data.

1. Under Control Stages, click the Math stage and drag it to the canvas, placing it where you want on the flow.
2. Connect the stage to other stages on the canvas.
3. Double-click the Math stage. The **Math Options** dialog box appears, with the Expressions tab open. This view shows the input fields, the Calculator, and the Expressions canvas. Alternately, you can click the Functions tab to use functions instead of the Calculator.

The Input fields control lists the valid fields found on the input port. Field name syntax is very flexible but has some restrictions based on Groovy scripting rules. If you are not familiar with Groovy scripting, see this website for complete information about Groovy: groovy-lang.org.

Note: This stage is not available in the tech preview version of Flow Designer.

Using the Calculator

The Calculator control contains buttons for entering numeric constants and operators into an expression. Double-clicking fields, constants, operators, and functions will insert them into an expression.

Table 5: Calculator Operators

Description

Used to go back one space in an expression
Pi, a mathematical constant which is the ratio of the circumference of a circle to its diameter
Euler's Number, a mathematical constant that is the base of the natural logarithm
Division
Multiplication
Addition
Subtraction
Power of (for example, x^2 is x to the power of 2)
Modulo, the remainder of an operation
Semicolon, used at the end of expressions
Assignment operator
Parentheses, to represent hierarchy in an expression
Decimal point
Conditional statement to take action if a condition is true, otherwise, take a different action

Description

Multiple conditional statement to take action if a condition is true, otherwise, take a different action

Equal to, in a math function

Not equal to

Logical and

Logical or

Greater than

Greater than or equal to

Less than

Less than or equal to

Using Functions and Constants

The Math stage provides several functions that can be used in an expression. Functions take the general form `function(parameter)`; `function(parameter, parameter)`; `function(parameter, ...)`, where "parameter" is a numeric constant, a variable, or a math expression. Functions can be used with other math expressions (for example: $x = \sin(y) * \cos(z)$).

Constants, Conversion, Math, and Trigonometry

Each of the supported functions is listed below within its corresponding category.

Table 6: Supported Functions

Function	Description
Constants	
e	A mathematical constant that is the base of the natural algorithm.

Function	Description
false	A Boolean constant that represents the value false.
Infinity	A mathematical constant that represents infinity.
NaN	A mathematical constant that represents a value that is not a number.
Pi	A mathematical constant that is the ratio of the circumference of a circle to its diameter.
true	a Boolean constant that represents the value true.
Conversion	
Abs (value)	Takes one parameter. Returns the absolute value of the given value.
Ceil (value)	Takes one parameter. Returns a rounded-up value (for example, Ceil(5.5) returns 6).
DegToRad (value)	Takes one parameter. Converts a given value from degrees to radians.
Floor (value)	Takes one parameter. Returns a rounded-down value (for example, Floor(5.5) returns 5).
RadToDeg (value)	Takes one parameter. Converts a given value from radians to degrees.
Round (value)	Takes one parameter. Returns a rounded value.
Math	
Avg (value, value,...)	Takes one or more parameters. Returns the average of all given values.

Function	Description
Exp (value)	Takes one parameter. Returns Euler's number raised to the power of the value.
Fac (value)	Takes one parameter. Returns the factorial of a given value (for example, Fac(6) is computed to $6*5*4*3*2*1$ and returns 720).
Ln (value)	Takes one parameter. Returns the natural logarithm (base e) of a given value.
Log (value)	Takes one parameter. Returns the natural logarithm (base 10) of a given value.
Max (value, value,...)	Takes one or more parameters. Returns the maximum value passed in.
Min (value, value,...)	Takes one or more parameters. Returns the minimum value passed in.
Sqrt (value)	Takes one or more parameters. Returns the square root of the value passed in.
Sum (value)	Takes one parameter. Returns the sum of the given values.
Trigonometry	
ArcCos (value)	Takes one parameter. Returns the arc cosine of a value.
ArcSin (value)	Takes one parameter. Returns the arc sine of a value.

Function	Description
ArcTan (value)	Takes one parameter. Returns the arc tangent of a value.
Cos (value)	Takes one parameter. Returns the cosine of a value.
Ln (value)	Takes one parameter. Returns the natural logarithm (base e) of a given value.
Sin (value)	Takes one parameter. Returns the sine of a value.
Tan (value)	Takes one parameter. Returns the tangent of a value.

Using Conditional Statements

Conditional statements can be used to take actions depending on whether various conditions evaluate to true or false. Grouping using parentheses (and) can be used for more complex conditions.

Table 7: Conditions

Condition	Description
Equals	expression == expression
Not Equals	expression != expression
Greater Than	expression > expression
Greater Than or Equal To	expression >= expression
Less Than	expression < expression

Condition	Description
Less Than or Equal To	expression <= expression
Not condition	!condition
And	condition && condition
Or	condition condition

If Statement

```
if(condition)
{
    actions to take if condition is true
}
```

Brackets are needed only if more than one statement is run after the "if."

If-Else If Statements

```
if(condition)
{
    actions to take if condition is true
}
else if(condition)
{
    actions to take if condition is true
}
else if...
```

```
if(SideLength != NaN)
{
    AreaOfPolygon=
    ((SideLength^2)*NumberOfSides)/
    (4*Tan(pi/NumberOfSides));
}
else if(Radius != NaN)
{
    AreaOfPolygon=
    (Radius^2)*NumberOfSides*Sin((2*pi)/NumberOfSides)/2;
}
```

One or more else if statements can be specified. Brackets are needed only if more than one statement is run after the "if-else- if-else."

Else-If Statement

```

if(condition)
{
    actions to take if condition is true
}
else if(condition)
{
    actions to take if condition is true
}
else if...
else
{
    actions to take if no conditions are met
}

```

Using the Expressions Console

The Expressions console is used to enter math expressions to be evaluated by the Math stage. The Input, Calculator, and Functions controls are used to insert values into this console. You can also manually type expressions into the console. Expressions take the form of a constant, variable, or math operation, and consist of numeric constants and variables. Numeric constants are whole or decimal numbers, which can be signed. Variables represent data from the incoming row. For example, if fields x, y, and z are defined in the input, then x, y, and z can be used in an expression. Variables are replaced with field values at runtime.

The Math stage also allows grouped expressions, which involve using parentheses to group expressions and override operator precedence. For example, $2*5^2$ equals 50, while $(2*5)^2$ equals 100.

Note: Every expression you enter must end with a semi-colon.

Additionally, conditional statements can be used in the Expressions console to take actions depending on whether various conditions evaluate to true or false. See [Using Conditional Statements](#) on page 63 for more information about conditional statements.

The Math stage deals primarily with assignment expressions, in which the output from an expression is assigned to a variable. Multiple assignment operations are supported in the stage and can use the output of a previous assignment operation.

Assignment Expression Examples

In the scenario below, $x=10$ and $z=1000$:

```

x=5+5
z=x*100

```

In the scenario below, the area of a polygon is calculated based on the length of one side and the number of sides.

```
AreaOfPolygon=
  ((SideLength^2)*NumberOfSides) /
  (4*Tan(pi/NumberOfSides));
```

Using the Fields Control

The Fields control allows you to change input and output field types. You can change field types from within this control by clicking the drop-down arrow in the Type column and selecting from the list, which includes these options:

boolean A logical type with two values: true and false. Boolean variables can be used in conditional statements to control flow. The code sample, below, shows a Boolean expression:

```
if(x && y)
  z=1;
else if(x)
  z=2;
else if(y)
  z=3;
else
  z=4;
```

double A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. 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 $(2-2^{-23}) \times 2^{127}$. 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).

Using the Preview Control

The Preview control allows you to test math expressions. Fields are listed in the Input Data area; you can provide specific values to pass to the expression and view the output in the Results area beneath Input Data.

Numeric fields are initialized to 0 (0.000 for double) and boolean fields are initialized to False. Double and float fields are limited to four decimal places, and integer and long fields have no decimal places.

Record Combiner

Record Combiner merges two or more records from multiple streams into a single record based on a commonality. Record Combiner can have one or more stage input ports. For example, you can have one group of records from one stage input (port) and the other group from a second stage input (port 2), and the records will merge into a single record. If you delete a middle stage, the ports will not renumber consecutively.

Note: Record Combiner will not release a record on output until each of its input ports has received a record. It must combine as many records as it has input ports before outputting a record.

You can specify which port should be preserved in cases where the input streams have fields of the same name. For example, if you are combining records from two streams, and both streams contain a field named AccountNumber, you could specify which stream's AccountNumber field you want to preserve by choosing the Record Combiner input port that corresponds to the stream you want to preserve. The data from the AccountNumber field in the other stream would be discarded.

Record Joiner

Record Joiner performs a SQL-style `JOIN` operation to combine records from different streams based on a relationship between fields in the streams. You can use **Record Joiner** to join records from multiple files, multiple databases, or any upstream channels in the flow. You must connect at least two input channels to **Record Joiner**. The results of the `JOIN` operation are then written to one output channel. Optionally, records that do not match the join condition can be written to a separate output channel.

Using Record Joiner

To use the **Record Joiner** stage in a new Flow, perform these steps:

1. On the **Spectrum Flow Designer** Home page, click **New**.
2. On the **New Flow** page, click **Job**, **Service**, or **Subflow**, as required and then click the corresponding blank canvas.
3. Click **Ok**.
4. In the dialog box that appears, give a name to the **Flow**, **Job**, **Service**, or **Subflow** you are creating.
5. Click **Ok**.
6. From the **Palette Panel** drag the **Record Joiner** stage to the canvas.

Note: Record Joiner is one of the **Control Stages**.

7. Drag all **Sources** (of the records that are to be joined) to the canvas and connect their **Output Port** to the **Record Joiner Input Port**.
8. Drag the **Sink** for the joined records and connect its **Input Port** to the **Record Joiner Output Port**.
9. Configure the Sources. See the documentation of the respective stage for field-level details.
10. Click **Record Joiner** and configure the **Join Definition** as described below.

Join Definition

Option	Description
Left port	<p>The port whose records you want to use as the left table in the <code>JOIN</code> operation. All other input ports will be used as right tables in the <code>JOIN</code> operation.</p> <p>Note: "Left" table and "right" table are SQL <code>JOIN</code> concepts. Before using Record Joiner you should have a good understanding of the SQL <code>JOIN</code> operation. For more information, see wikipedia.org/wiki/Join_(SQL).</p>
Join type	<p>The type of <code>JOIN</code> operation you want to perform. One of the following:</p> <p>Left Outer Returns all records from the left port even if there are no matches between the left port and the other ports. This option returns all records from the left port plus any records that match in any of the other ports.</p> <p>Full Returns all records from all ports.</p> <p>Inner Returns only those records that have a match between the left port and another port. For example, if you have four input sources and port 1 is the left port, an inner join will return records that have matching fields between port 1 and port 2, port 1 and port 3, and port 1 and port 4.</p>
Join Fields	<p>The field or fields from the left port that must match the data in a field from another port in order for the records to be joined.</p> <p>Note: The valid data types for join fields are integer, string, datetime, date, long, float, double, and big decimal.</p>

Option	Description
Data from the left port is sorted	<p>Specifies whether the records in the left port are already sorted by the field specified in Join Fields. If the records are already sorted, checking this box can improve performance. If you do not check this box, Record Joiner will sort the records according to the field specified in Join Fields before performing the join operation.</p> <p>If you have specified multiple join fields, then the records must be sorted using the order of the fields listed in Join Fields. For example, if you have two join fields:</p> <p style="padding-left: 40px;">Amount Region</p> <p>Then the records must be sorted first by the Amount field, then by the Region field.</p> <p>Important: If you select this option but the records are not sorted, you will get incorrect results from Record Joiner. Only select this option if you are sure that the records in the left port are already sorted.</p>
Join Definitions	<p>Describes the join conditions that will be used to determine if a record from the left port should be joined with a record from one of the other ports: <code>port1.Name = port2.Name</code></p> <p>This indicates that if the value in the Name field of a record from port1 matches the value in the Name field of a record from port2, the two records will be joined.</p> <p>To modify a join condition, click Modify. Select a field from the right port whose data must match the data in the join field from the left port in order for the records to be joined. If you want to change the left port field, click Cancel and change it in the Join Fields field. If the records in the right port are sorted by the join field, check the box Data from the right port is sorted. Checking this box can improve performance.</p> <p>Important: If you select Data from the right port is sorted but the records are not sorted, you will get incorrect results from Record Joiner. Only select this option if you are sure that the records in the right port are already sorted.</p>

Field Resolution

This tab specifies which port's data to use in the joined record in cases where the same field name exists in more than one input port. For example, if you are performing a join on two sources of data, and each source contains a field named DateOfBirth, you can specify which port's data to use in the DateOfBirth field in the joined record.

If there are fields of the same name but with different data, and you want to preserve both fields' data in the joined record, you must rename one of the fields before the data is sent to Record Joiner. You can use the Transformer stage to rename fields.

Handling Records That Are Not Joined

In order for a record to be included in the Record Joiner output it must meet the join condition, or a join type must be selected that returns both joined records and those that did not meet the join condition. For example, a full join will return all records from all input ports regardless of whether a record meets the join condition. In the case of a join type that does not return all records from all ports, such as a left outer join or an inner join, only records that match the join condition are included in the Record Joiner output.

To capture the records that are not included in the result of the join operation, use the **not_joined** output port. The output from this port contains all records that were not included in the regular output port.

Records that come out of this port have the field **InputPortIndex** added to them. This field contains the number of the Record Joiner input port where the record came from. This allows you to identify the source of the record.

Note:

- For optimal performance of this stage, ensure two independent streams of records are joined to generate a consolidated output.
- If a single path is first branched using either a broadcaster or conditional router then re-joined back using a Record Joiner, the flow may hang. In case multiple stages are used between branching and joining, use the Sorter as close to the Record Joiner as possible.

Math

The Math stage handles mathematical calculations on a single data row and allows you to conduct a variety of math functions using one or more expressions. Data is input as strings but the values must be numeric or Boolean, based on the type of operation being performed on the data.

1. Under Control Stages, click the Math stage and drag it to the canvas, placing it where you want on the flow.
2. Connect the stage to other stages on the canvas.
3. Double-click the Math stage. The **Math Options** dialog box appears, with the Expressions tab open. This view shows the input fields, the Calculator, and the Expressions canvas. Alternately, you can click the Functions tab to use functions instead of the Calculator.

The Input fields control lists the valid fields found on the input port. Field name syntax is very flexible but has some restrictions based on Groovy scripting rules. If you are not familiar with Groovy scripting, see this website for complete information about Groovy: groovy-lang.org.

Note: This stage is not available in the tech preview version of Flow Designer.

Sorter

Sorter sorts records using the fields you specify. For example, you can have records sorted by names, cities, or any other field in your dataflow.

Sorting Records with Sorter

The Sorter stage allows you to sort records using the fields you specify.

1. Under Control Stages, drag Sorter to the canvas, placing it where you want on the dataflow.
2. Double-click Sorter.
3. Click **Add**.
4. Click the down-arrow in the **Field Name** column and select the field that you want to sort on.

Note: The list of available fields is based on the fields used in the previous stages in the dataflow.

5. In the **Order** column, choose whether you want to sort in ascending or descending order.
6. In the **Type** column, select the field's data type.

Note: If your incoming data is not in string format, the **Type** column will be disabled.

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.
double	A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. 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 $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -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.

7. To remove blank space from before and after the value before sorting, check the box in the **Trim** column. The trim option does not modify the value of the field. It only trims the value for the

purpose of sorting. Note that if your incoming data is not in string format, the **Trim** column will be disabled.

- In the **Treat Null As** column, select **Largest** or **Smallest** to indicate the placement of null values in the sorted list. The placement depends on the combination of options selected in the **Order** and **Treat Null As** fields, as shown in the table below:

Order	Treat Null As	Placement of null values in the sorted list
Ascending	Largest	Bottom of the list
Ascending	Smallest	Top of the list
Descending	Largest	Top of the list
Descending	Smallest	Bottom of the list

- Repeat until you have added all the fields you want to sort.
- Rearrange the sort order as desired by clicking **Up** or **Down**. This allows you to sort first by one field, then sort the resulting order again by another field.
- If you want to override the default sort performance options that have been defined by your administrator, click **Advanced**, check the **Override sort performance options** box, then specify these options:

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:

$$\frac{(NumberOfRecords \times 2)}{NumberOfTempFilesN} = InMemoryRecordLimit$$

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: $(InMemoryRecordLimit \times MaxNumberOfTempFiles \div 2) \geq TotalNumberOfRecords$

12. Click **OK**.

Note: You can remove the sort criteria as desired by highlighting a row and clicking **Remove**.

Splitter

A Splitter converts hierarchical data to flat data. Splitters have one input port and one output port that delivers data from the Splitter to the next stage. One way you could use the Splitter's functionality is to take a list of information in a file and extract each discrete item of information into its own data row. For example, your input could include landmarks within a certain distance of a latitudinal-longitudinal point, and the Splitter could put each landmark into a separate data row.

Using the Splitter Stage

1. Under Control Stages, click the **Splitter** and drag it to the canvas, placing it where you want on the flow and connecting it to input and output stages.
2. Double-click the Splitter. The **Splitter Options** dialog box appears.
3. Click the **Split at** drop-down to see other list types available for this stage. Click the list type you want the Splitter to create. The **Splitter Options** dialog box will adjust accordingly with your selection, showing the fields available for that list type.

Alternatively, you can click the ellipses (...) button next to the **Split at** drop-down. The **Field Schema** dialog box appears, showing the schema for the data coming into the Splitter. The list types are shown in bold, followed by the individual lists for each type. Also shown is the format of those fields (string, and double, for instance). Click the list type you want the Splitter to create and click **OK**. The **Splitter Options** dialog box will adjust accordingly with your selection, showing the fields available for that list type.

4. Click **Output header record** to return the original record with the split list extracted.

5. Click **Only when input list is empty** to return the original record only when there is no split list for that record.
6. Select which fields you want the Splitter to include on output by checking the **Include** box for those fields.
7. Click **OK**.

Splitter Example

This example takes output from a routing stage that includes driving directions and puts each direction (or list item) into a data row. The flow looks like this:



The flow performs the function as follows:

1. The **Read from File** stage contains latitudes, longitudes, and input key values to help you identify the individual points.

Read from File Options

File Properties | **Fields** | Sort Fields | Runtime

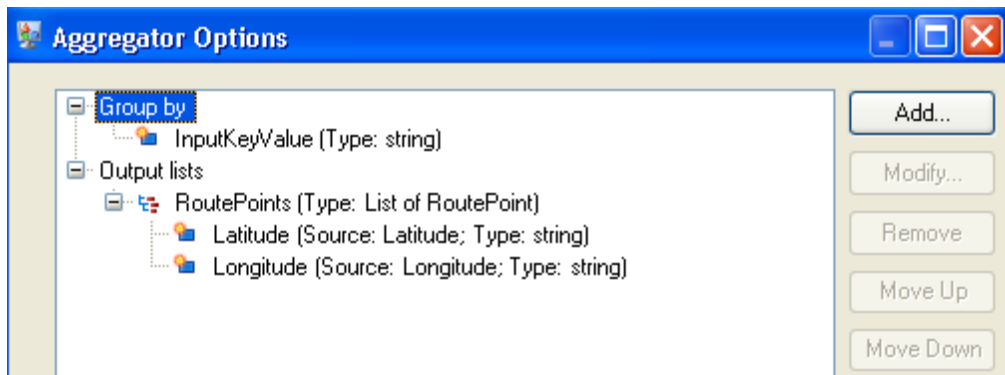
Name	Type	Position	<input type="checkbox"/>	Trim
InputKeyValue	string	1	<input type="checkbox"/>	
Latitude	string	2	<input type="checkbox"/>	
Longitude	string	3	<input type="checkbox"/>	

Buttons: Add..., Modify..., Remove, Regenerate

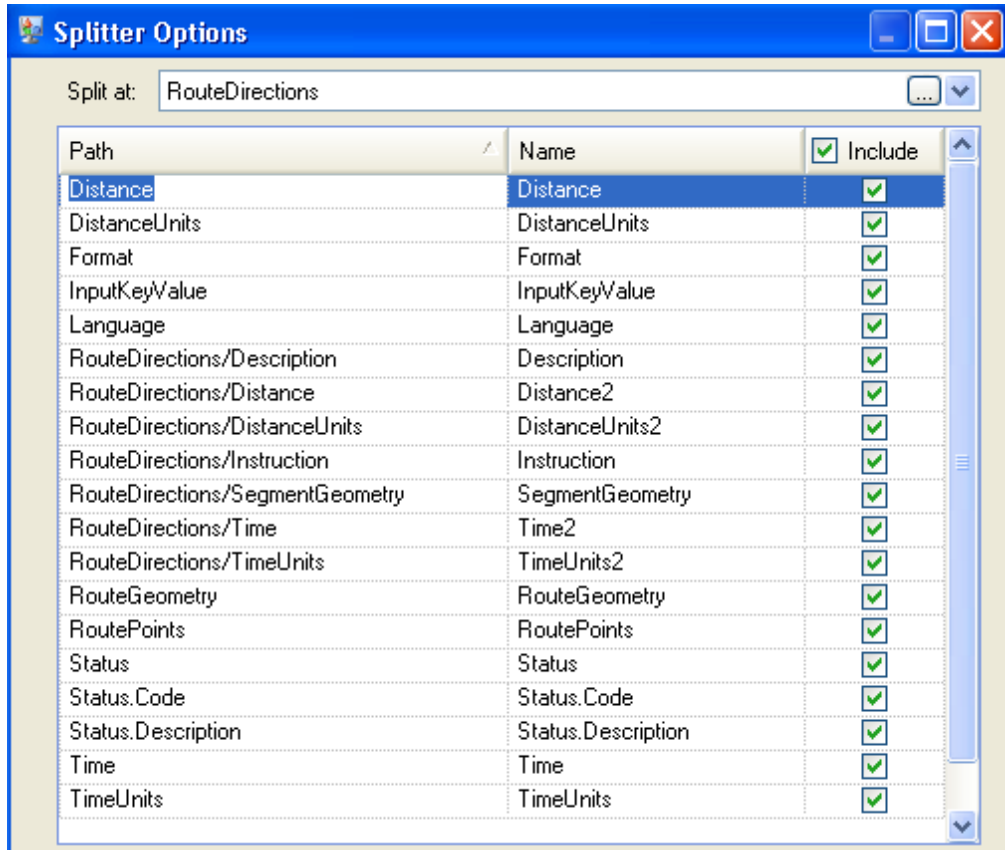
Preview

InputKeyValue	Latitude	Longitude
1	30.352549	-97.726456
1	32.938469	-96.598656
2	42.798782	-73.926880
2	33.413521	-117.605843

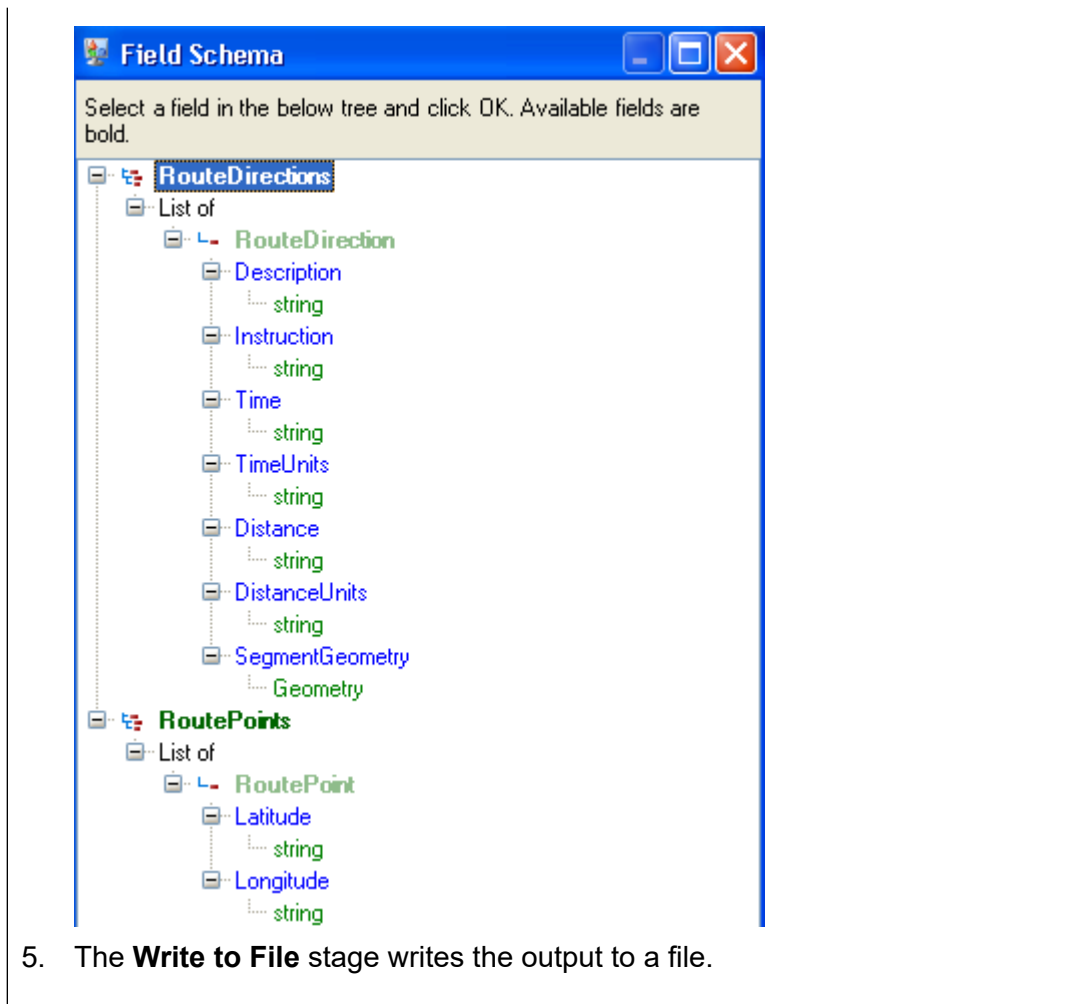
2. The Aggregator stage builds up the data from the Read from File stage into a schema (a structured hierarchy of data) and identifies the group of latitudes and longitudes as a list of route points, which is a necessary step for the next stage to work correctly.



3. Spectrum Spatial **Get Travel Directions** stage creates directions from one location to another using the route points from step 2.
4. The **Splitter** stage establishes that the data should be split at the Route Directions field and that the output lists should include all of the possible fields from the Get Travel Directions stage.



The schema is structured as follows, with Route Directions and Route Points being the available list types for this job:



Stream Combiner

Stream Combiner joins two or more streams of records from multiple stages. Stream Combiner has one or more stage input ports. For example, you can have one group of records from one stage and another group from a second stage, and the records will merge into a single stream.

Stream Combiner has no settings.

Transformer stage transform types

The Transformer stage modifies field values and formatting. You can perform more than one transform on a field as long as the input and output field names are identical.

General Transforms

Construct Field Uses values from existing fields and/or constant values to either replace field values or create a new field. For example, say you have a field named City and you want to add the phrase "City of" to the values in the City field. You would create a template like this:

```
City of ${City}
```

In the **To field** field, you would select the City field. This has the effect of replacing the existing values in the City field with a value constructed using the template. For example, if the value in the City field is Chicago, the new value would be City of Chicago.

Some characters must be preceded by a backslash to produce a valid template. For example, the single quote character must be preceded by a backslash like this: \'. See groovy-lang.org/syntax.html for a list of characters that must be escaped with a backslash.

Copy Copies the value from one field to another.

Custom Allows you to define your own transform using the Groovy language. For more information, see [Creating a Custom Transform](#) on page 81.

Rename Changes the name of a field. You can select from a list of field names already in the dataflow or you can type the name you want.

Status Changes the Status field to a value of either Success or Fail. When set to Fail, an optional Description and Code may also be set.

Formatting Transforms

Case Changes casing upper or lower case.

Mask Applies or removes characters from a field. For more information, see [Using a Mask Transform](#) on page 88.

Pad Adds characters to the left or right of the field value.

String Transforms

Minimize Whitespace Removes white space at the beginning and end of the field. It also replaces any sequence of white spaces (such as multiple, consecutive spaces) to a single white space character.

Remove Substring Removes all occurrences of a string from a field. For example, you could remove "CA" from the StateProvince field.

Substring Copies a contiguous sequence of characters from one field to another.

Trim Removes specified characters from the left, right, or both sides of a field. Note that this transform is case-sensitive.

Truncate Removes a specified number of characters from the left and right sides of a field.

List Transforms

This feature helps you to create canned transformation that operate on lists, for example input from read from XML.

For defining list transformations, follow these steps:

1. Select a list transformation operation. Input fields appear in a tree view on the right.
2. Select a valid field in the tree to apply the operation on. Properties for the operation show up below the input fields tree view.
3. Specify the operation properties and click add. The transform gets added to the list in the parent window, that is the 'Transformer Options' window.

Create Field Allows creating a field under the user selected list type field. For example if a list called Football has two clubs, Knitters and Lambs, you user can add a new club called Irons, for a total of three clubs.

Sort Performs sorting on values present in the selected field. In a complex list, the user needs to specify the key element for sorting while in case of simple list, the sorting takes place on the elements present in the list. The user can select the sort order as either ascending or descending. In the example of Football, when the list has three clubs, the user needs to select field 'name' under 'club' to sort the clubs based on name. The current club entries list as Irons, Knitters and Lambs if sort order is ascending and descending for sort order descending. Now, if the user wants the list of players sorted, the field 'player' needs to be selected and sort order defined for it

Sum Performs summation of all the values present in the selected field. The output is stored in a field specified by the user. For example, if the user wants to view the total points gained by each football club, user needs to select field 'points' under 'Tournament' and specify the output field name.

Copy Performs the copy operation from the selected field to the field specified by the user. When user selects a field to copy, the field and all fields under it (if any) are copied to the new field specified. This operation takes place at the same level of hierarchy.

Rename Performs the rename operation of the selected field to the new name specified by the user.

This sample XML code provides a reference to the List Transform feature:

```
<?xml version="1.0"?>
<sports_details>
  <sports name="football">
    <clubs>
      <club name="Knitters">
```

```

    <player>Samuel</player>
    <player>Messi</player>
    <player>kaka</player>
    <player>Alan</player>
    <coach>Stuart</coach>
    <Tournament name="Football League">
<result>won</result>
<points>4</points>
    </Tournament>
    <Tournament name="UEFA">
<result>draw</result>
<points>2</points>
    </Tournament>
</club>
<club name="Lambs">
    <player>Ronaldo</player>
    <player>Neymar</player>
    <player>Zlatan</player>
    <player>Mesut</player>
    <coach>Ivan</coach>
    <Tournament name="Airtel League">
<result>draw</result>
<points>2</points>
    </Tournament>
    <Tournament name="Champions League">
<result>lost</result>
<points>0</points>
    </Tournament>
</club>
<club name="Irons">
    <player>Scott</player>
    <player>Paul</player>
    <player>John</player>
    <player>Andrew</player>
    <coach>Jeff</coach>
    <Tournament name="CAF">
<result>won</result>
<points>4</points>
    </Tournament>
    <Tournament name="Copa America">
<result>won</result>
<points>4</points>
    </Tournament>
</club>
</clubs>
</sports>
<sports name="badminton">
<clubs>
    <club name="Shuttlers">
    <player>Saina</player>
    <player>Viktor</player>
    <player>Chen</player>
    <player>Srikanth</player>

```

```

    <coach>Jan</coach>
    <Tournament name="Olympic Games">
<result>won</result>
<points>4</points>
    </Tournament>
    <Tournament name="Commonwealth Games">
<result>won</result>
<points>4</points>
    </Tournament>
</club>
<club name="Choppers">
    <player>Wang</player>
    <player>Sindhu</player>
    <player>Carolina</player>
    <player>Li Xuerui</player>
    <coach>Ratchanok</coach>
    <Tournament name="World Junior">
<result>draw</result>
<points>2</points>
    </Tournament>
    <Tournament name="Uber Cup">
<result>draw</result>
<points>2</points>
    </Tournament>
</club>
<club name="Lobbers">
    <player>Nozomi</player>
    <player>Chou</player>
    <player>Marc</player>
    <player>Lin</player>
    <coach>Kevin</coach>
    <Tournament name="World Senior">
<result>won</result>
<points>4</points>
    </Tournament>
    <Tournament name="Thomas Cup">
<result>won</result>
<points>4</points>
    </Tournament>
</club>
</clubs>
</sports>
</sports_details>

```

Changing the Order of Transforms

If you have more than one transform to run on a particular output field, you can define the order in which they run.

Note: If you map a single field to two different output fields (for example, `ValidateAddress.City` to `Output.City1` and `ValidateAddress.City` to

`Output.City2`), and you add transforms to each field, the transform for the secondary field must be run first. You must change the order of the transforms to run the second field transform (`Output.City2`) first.

1. Double-click the Transformer stage. The Transformer Options dialog box appears.
2. Select a transform and use the Move Up and Move Down buttons to rearrange the order of the transforms. Spectrum Flow Designer runs the top transform first.

Note: Dependent transforms cannot be moved above primary transforms (the transforms upon which the dependent transforms rely).

3. Click **OK**.

Creating a Custom Transform

The Transformer stage has predefined transforms that perform a variety of common data transformations. If the predefined transforms do not meet your needs, you can write a custom transform script using Groovy. This procedure describes how to create basic custom transforms using Groovy. For complete documentation on Groovy, see groovy-lang.org.

1. In Spectrum Enterprise Designer, add a Transformer stage to the dataflow.
2. Double-click the Transformer stage.
3. Click **Add**.
4. Under **General**, click **Custom**.
5. In the **Custom transform name** field, enter a name for the transform you will create. The name must be unique.
6. Click **Script Editor**.

This editor provides a variety of features to make developing your transform easier, such as code completion and palettes listing functions and fields.

Task	Instructions
To add a function	<p>In the Functions pane, double-click the function you want to add.</p> <p>Note: The functions listed in the editor are functions provided to make writing custom transform scripts easier. They perform functions that would otherwise require multiple lines of Groovy code to accomplish. They are not standard Groovy functions.</p>
To get the value from a dataflow field	<p>In the Input Fields pane, double-click the input field you want. The following will be added to your script:</p> <pre>data['FieldName']</pre>

Task	Instructions
	<p>For example, if you want to get the value from the field <code>CurrentBalance</code>, the following would be added:</p> <pre data-bbox="488 394 1425 464">data['CurrentBalance']</pre>
<p>To set the value of a dataflow field</p>	<p>Enter this code in the script editor:</p> <pre data-bbox="488 594 1425 663">data['FieldName']=NewValue</pre> <p>For example, to set the field <code>Day</code> to the day of the week contained in the field <code>PurchaseDate</code>:</p> <pre data-bbox="488 768 1425 800">data['Day']=dayOfWeek(data['PurchaseDate'])</pre> <p>In this example, the function <code>dayOfWeek()</code> is used to get the day from the date value in the <code>PurchaseDate</code> field, and the result is written to the <code>Day</code> field.</p> <p>Tip: You can double-click the name of the output field in the Output Fields pane to add the field reference to the script.</p>
<p>To change the scope of a script variable in a dataflow</p>	<p>To change the scope of a script variable from a single input record to all the input records in a dataflow, use the <code>@Field</code> annotation in your script as shown:</p> <pre data-bbox="488 1224 1425 1304">import groovy.transform.Field; @Field ['data type']['VariableName']= Value;</pre> <p>For example, to set the scope of a variable <code>RecordNumber</code> to a single input record, specify this:</p> <pre data-bbox="488 1419 1425 1545">int recordNumber = 1; data['Record_Number']= recordNumber; recordNumber++;</pre> <p>The output will be:</p>

Task	Instructions																																												
	<div data-bbox="483 310 857 630"> <table border="1"> <thead> <tr> <th>Date</th> <th>Record_Number</th> </tr> </thead> <tbody> <tr><td>3/14/18</td><td>1</td></tr> <tr><td>3/15/18</td><td>1</td></tr> <tr><td>3/16/18</td><td>1</td></tr> <tr><td>3/17/18</td><td>1</td></tr> <tr><td>3/18/18</td><td>1</td></tr> <tr><td>3/19/18</td><td>1</td></tr> <tr><td>3/20/18</td><td>1</td></tr> <tr><td>3/21/18</td><td>1</td></tr> <tr><td>3/22/18</td><td>1</td></tr> <tr><td>3/23/18</td><td>1</td></tr> </tbody> </table> </div> <p data-bbox="483 655 1370 688">To change the scope of this variable to all input records, specify this:</p> <div data-bbox="500 714 1425 865"> <pre>import groovy.transform.Field @Field int recordNumber = 1; data['Record_Number'] = recordNumber; recordNumber++;</pre> </div> <p data-bbox="483 886 727 919">The output will be:</p> <div data-bbox="483 940 863 1260"> <table border="1"> <thead> <tr> <th>Date</th> <th>Record_Number</th> </tr> </thead> <tbody> <tr><td>3/14/18</td><td>1</td></tr> <tr><td>3/15/18</td><td>2</td></tr> <tr><td>3/16/18</td><td>3</td></tr> <tr><td>3/17/18</td><td>4</td></tr> <tr><td>3/18/18</td><td>5</td></tr> <tr><td>3/19/18</td><td>6</td></tr> <tr><td>3/20/18</td><td>7</td></tr> <tr><td>3/21/18</td><td>8</td></tr> <tr><td>3/22/18</td><td>9</td></tr> <tr><td>3/23/18</td><td>10</td></tr> </tbody> </table> </div>	Date	Record_Number	3/14/18	1	3/15/18	1	3/16/18	1	3/17/18	1	3/18/18	1	3/19/18	1	3/20/18	1	3/21/18	1	3/22/18	1	3/23/18	1	Date	Record_Number	3/14/18	1	3/15/18	2	3/16/18	3	3/17/18	4	3/18/18	5	3/19/18	6	3/20/18	7	3/21/18	8	3/22/18	9	3/23/18	10
Date	Record_Number																																												
3/14/18	1																																												
3/15/18	1																																												
3/16/18	1																																												
3/17/18	1																																												
3/18/18	1																																												
3/19/18	1																																												
3/20/18	1																																												
3/21/18	1																																												
3/22/18	1																																												
3/23/18	1																																												
Date	Record_Number																																												
3/14/18	1																																												
3/15/18	2																																												
3/16/18	3																																												
3/17/18	4																																												
3/18/18	5																																												
3/19/18	6																																												
3/20/18	7																																												
3/21/18	8																																												
3/22/18	9																																												
3/23/18	10																																												
<p data-bbox="212 1339 477 1444">To create a new field using a numeric data type</p>	<p data-bbox="483 1339 928 1373">Enter this code in the script editor:</p> <div data-bbox="500 1394 1425 1453"> <pre>data['FieldName'] = new constructor;</pre> </div> <p data-bbox="483 1474 932 1507">Where <i>constructor</i> is one of these:</p> <p data-bbox="483 1528 847 1562">java.lang.Double(<i>number</i>)</p> <p data-bbox="669 1570 1214 1604">Creates a field with a data type of Double.</p> <p data-bbox="483 1625 818 1659">java.lang.Float(<i>number</i>)</p> <p data-bbox="669 1667 1185 1701">Creates a field with a data type of Float.</p> <p data-bbox="483 1722 847 1755">java.lang.Integer(<i>number</i>)</p> <p data-bbox="669 1764 1370 1835">Creates a field with a data type of Integer. You can also create a new integer field by specifying a whole number.</p>																																												

Task	Instructions
	<p>For example, this will create an integer field with a value of 23:</p> <pre data-bbox="670 401 1365 457">data['MyNewField'] = 23;</pre> <p>java.lang.Long(<i>number</i>) Creates a field with a data type of Long.</p> <p>For example, to create a new field named "Transactions" with a data type of Double and the value 23.10, you would specify the following:</p> <pre data-bbox="488 667 1422 699">data['Transactions'] = new com.java.lang.Double(23.10);</pre>
<p>To create a new field using a date or time data type</p>	<p>Enter this code in the script editor:</p> <pre data-bbox="488 856 1422 905">data['FieldName'] = new constructor;</pre> <p>Where <i>constructor</i> is one of these:</p> <p>com.pb.spectrum.api.datetime.Date(<i>year,month,day</i>) Creates a field with a data type of date. For example, December 23, 2013 would be:</p> <pre data-bbox="670 1115 1365 1171">2013,12,23</pre> <p>com.pb.spectrum.api.datetime.Time(<i>hour,minute,second</i>) Creates a field with a data type of time. For example, 4:15 PM would be:</p> <pre data-bbox="670 1329 1365 1386">16,15,0</pre> <p>com.pb.spectrum.api.datetime.DateTime(<i>year,month,day,hour,minute,second</i>) Creates a field with a data type of DateTime. For example, 4:15 PM on December 23, 2013 would be:</p> <pre data-bbox="670 1596 1365 1652">2013,12,23,16,15,0</pre> <p>For example, to create a new field named "TransactionDate" with a data type of Date and the value December 23, 2013, you would specify this:</p> <pre data-bbox="488 1770 1422 1839">data['TransactionDate'] = new com.pb.spectrum.api.datetime.Date(2013,12,23);</pre>

Task	Instructions
<p>To create a new field with a data type of Boolean</p>	<p>Enter this code in the script editor:</p> <pre data-bbox="500 386 1425 449">data['FieldName'] = true or false;</pre> <p>For example, to create a field named <code>IsValidated</code> and set it to <code>false</code>, you would specify this:</p> <pre data-bbox="500 554 1425 617">data['IsValidated'] = false;</pre>
<p>To create a new list field</p>	<p>Use the <code>factory.create()</code> method to create new fields in a record then use the leftShift operator <code><<</code> to append the new record to the list field.</p> <pre data-bbox="500 806 1425 1255">NewListField = [] NewRecord = factory.create() NewRecord['NewField1'] = "Value" NewRecord['NewField12'] = "Value" ... NewListField << NewRecord NewRecord = factory.create() NewRecord['NewField1'] = "Value" NewRecord['NewField12'] = "Value" ... NewListField << NewRecord data['ListOfRecords'] = NewListField</pre> <p>For example, this creates a new list field called "addresses" consisting of two "address" records.</p> <pre data-bbox="500 1373 1425 1738">addresses = [] address = factory.create() address['AddressLine1'] = "123 Main St" address['PostalCode'] = "12345" addresses << address address = factory.create() address['AddressLine1'] = "PO Box 350" address['PostalCode'] = "02134" addresses << address data['Addresses'] = addresses</pre>

Task	Instructions
	<p>You can also create a new list field that contains a list of individual fields rather than a list of records. For example, this creates a new list field called PhoneNumbers containing home and work phone numbers:</p> <pre data-bbox="488 436 1422 604"> phoneNumbers = [] phoneNumbers << data['HomePhone'] phoneNumbers << data['WorkPhone'] data['PhoneNumbers'] = phoneNumbers </pre>
<p>To concatenate fields</p>	<p>Use the + symbol. For example, this example concatenates the FirstName field and the LastName field into a value and stores it in the FullName field</p> <pre data-bbox="488 762 1422 888"> String fullname = data['FirstName'] + ' ' + data['LastName']; data['FullName']=fullname; </pre> <p>In this example there are two input fields (AddressLine1 and AddressLine2) which are concatenated and written to the output field Address.</p> <pre data-bbox="488 999 1422 1125"> address1 = data['AddressLine1']; address2 = data['AddressLine2']; data['Address']=address1+ ',' + address2; </pre>
<p>To parse a field</p>	<p>Identify a separation character then use <code>substring</code> to parse the field. In this example, if the PostalCode field is greater than five characters, it separates the five-character ZIP Code and the +4 portion and writes them to separate fields in the output record.</p> <pre data-bbox="488 1371 1422 1770"> if (data['PostalCode'].length() > 5) { String postalCode = data['PostalCode']; int separatorPosition = postalCode.indexOf('-'); String zip = postalCode.substring(0, separatorPosition); String plusFour = postalCode.substring(separatorPosition + 1, postalCode.length()); data['Zip']=zip; data['PlusFour']=plusFour; } </pre>

Task	Instructions
To perform conditional processing	<p>Use an <code>if</code> or <code>switch</code> statement. These are the most common conditional processing constructs. For more information see groovy-lang.org.</p> <p>This example sets the field <code>AddressCity</code> to the first address line and city name if the city is Austin.</p> <pre>city = data['City']; address1 = data['AddressLine1'] if(city.equals('Austin')) data['AddressCity']=address1 + ',' + city;</pre>
To perform looping	<p>Use the <code>for</code> loop. This is the only looping construct you should need. For more information about looping or syntax see groovy-lang.org.</p>
To augment data	<p>Define a constant and use the concatenation character <code>+</code>. For example, this script appends the word "Incorporated" to the <code>FirmName</code> field.</p> <pre>firmname = data['FirmName']; constant = 'Incorporated'; if(firmname.length() > 0) data['FirmName']=firmname + ' ' + constant;</pre>
To access an option specified at runtime	<p>If the dataflow has runtime options enabled, you can access settings passed to the dataflow at runtime by using this syntax:</p> <pre>options.get("optionName")</pre> <p>For example, to access an option named <code>casing</code>, you would include this in your custom transform script:</p> <pre>options.get("casing")</pre>

7. After you are done entering your script, click the "X" button in the window to close the editor.
8. In the **Input fields** field, select the field or fields to which you want to apply the transform.
9. In the **Output fields** field, specify the field to which you want to write the output from the transform. If necessary, you can define a new field by clicking the **Add** button to the right of the **Output fields** field.
10. When you are done, click the **Add** button at the bottom of the window.

- Click **OK**.

Using a Mask Transform

You can use the Transformer stage to apply a mask transform to a field. A mask transform applies characters to a field, or removes characters from a field, using a specified pattern. For example, using a mask transform you could format a string of numbers like 8003685806 into a phone number like this: (800) 368 5806.

- In Spectrum Enterprise Designer, drag a Transformer stage to the canvas and connect it in the desired location.
- Double-click the Transformer stage.
- Click **Add**.
- Expand **Formatting** and select **Mask**.
- Select the type of mask you want to use.

Apply Adds characters to a field to form the string into a new pattern.

Remove Extracts a pattern of characters from a string.

- In the **Mask string** field, specify the pattern you want to use when either adding characters or removing characters.

There are two types of characters you use when specifying the mask string: literal characters and mask characters.

Literal characters represent actual characters that are present in the string. When a remove mask is used, the input character must match the literal character exactly. If that is the case, then they will be removed from the input. Similarly, the literal characters will be added to the input in the position indicated by the mask definition when the apply mask is used.

The other type of character you can use in a mask string is a mask character. A mask character indicates the type of character that can be in a particular location of the input string. For example, if you have an input where the first character is a number, the first mask character needs to be #. Anything in the input that matches this mask character will be kept in the output.

The table below lists the mask characters you can use in the **Mask string** field:

Table 8: Mask Characters

Character	Definition
#	Any number.
'	Escape character, used to escape any of the special formatting characters.

Character	Definition
U	Any character. All lowercase letters are mapped to upper case.
L	Any character. All upper case letters are mapped to lower case.
A	Any character or number.
?	Any character.
*	Anything.
H	Any hex character (0-9, a-f or A-F).

7. Click **Add**.
8. Click **OK**.

Mask Transform Examples

This is an apply mask that applies formatting to a string. Because "(" and ")" and <space> are literals, they will be added to the output. All the numbers will be kept because # is a mask character.

Input: 8003685806

Mask string: (###) ### ####

Output: (800) 368 5806

This remove mask example omits the dash in the ZIP Code.

Input: 60510-1135

Mask string: *****-*****

Output: 605101135

Unique ID Generator

The Unique ID Generator stage creates a unique key that identifies a specific record. A unique ID is crucial for data warehouse initiatives in which transactions may not carry all name and address

data, but must be attributed to the same record or contact. A unique ID may be implemented at the individual, household, business, and premises level. Unique ID Generator provides a variety of algorithms to create unique IDs.

The unique ID is based on either a sequential number or date and time stamp. In addition, you can optionally use a variety of algorithms to generate data to appended to the ID, thereby increasing the likelihood that the ID will be unique. The sequential number or date and time stamp IDs are required and cannot be removed from the generated ID.

Unique ID Generator can be used to generate a non-unique key using one of the key generation algorithms. In non-unique mode, you can create keys to use for matching. This may be useful in a data warehouse where you have already added keys to a dimension and you want to generate a key for new records in order to see if the new records match an existing record.

This example shows that each record in the input is assigned a sequential record ID in the output.

Record	RecordID
John Smith	0
Mary Smith	1
Jane Doe	2
John Doe	3

The Unique ID stage produces a field named RecordID which contains the unique ID. You can rename the RecordID field as required.

Defining a Unique ID

By default, the Unique ID Generator stage creates a sequential ID, with the first record having an ID of 0, the second record having an ID of 1, the third record having an ID of 2, and so forth. If you want to change how the unique ID is generated, follow this procedure.

1. In the Unique ID Generator stage, on the **Rules** tab, click **Modify**.
2. Choose the method you want to use to generate the unique ID.
For more information, see [Unique ID Definition Methods](#) on page 91.
3. Click **OK**.

Unique ID Definition Methods

Options	Description
Sequential Numeric tag starting at	<p>Assigns an incremental numeric value to each record starting with the number you specify. If you specify 0, the first record will have an ID of 0, the second record will have an ID of 1, and so on.</p> <p>Note: For this Unique Key, ensure you do not increase the Runtime instances (in the Runtime Performance tab) beyond 1 as this can create duplicate IDs.</p>

Options	Description
Sequential Numeric tag starting at value in a database field	<p>Assigns an incremental numerical value to each record starting with the maximum number read from the database field. This number is then incremented by 1 and assigned to the first record. For example, if the number read from the database field is 30, the first record will have an ID of 31, the second record will have an ID of 32, and so forth.</p> <p>Connection Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Spectrum Management Console. If you need to make a new database, or modify or delete an existing connection, click Manage.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection name Enter a name for the connection. This can be anything you choose.</p> <p>Database driver Select the appropriate database type.</p> <p>Connection options Specify the host, port, instance, user name, and password to use to connect to the database.</p> <p>Table view Specifies the table or view in the database that you want to query.</p> <p>Database field Select a column from the list to generate a unique key.</p> <p>The supported data types for unique ID generation are:</p> <p>long A numeric data type that contains both negative and positive whole numbers between -2^{63} ($-9,223,372,036,854,775,808$) and $2^{63}-1$ ($9,223,372,036,854,775,807$).</p> <p>integer A numeric data type that contains both negative and positive whole numbers between -2^{31} ($-2,147,483,648$) and $2^{31}-1$ ($2,147,483,647$).</p> <p>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.</p> <p>double A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is $-1.79769313486232E+308$ to $1.79769313486232E+308$.</p> <p>float A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is $-3.402823E+38$ to $3.402823E+38$.</p>

Options	Description
Date/Time stamp	Creates a unique key based on the date and time stamp instead of sequential numbering.
UUID	Creates a universally unique 32-digit identifier key for each record. The digits in the key are displayed in five groups separated by hyphens, in the form 8-4-4-4-12 for a total of 36 characters (32 alphanumeric characters and four hyphens). Example: 123e4567-e89b-12d3-a456-432255330000
Off	Select this option only if you want to generate a non-unique key using an algorithm.

Using Algorithms to Augment a Unique ID

Unique ID Generator generates a unique ID for each record by either numbering each record sequentially or generating a date-time stamp for each record. You can optionally use algorithms to append additional information to the sequential or date-time unique ID, thereby creating a more complex unique ID and one that is more likely to be truly unique.

1. In the Unique ID Generator stage, click **Add**.
2. In the **Algorithm** field, select the algorithm you want to use to generate additional information in the ID.

Consonant	Returns specified fields with consonants removed.
Double Metaphone	Returns a code based on a phonetic representation of their characters. Double Metaphone is an improved version of the Metaphone algorithm, and attempts to account for the many irregularities found in different languages.
Koeln	Indexes names by sound as they are pronounced in German. Allows names with the same pronunciation to be encoded to the same representation so that they can be matched, despite minor differences in spelling. The result is always a sequence of numbers; special characters and white spaces are ignored. This option was developed to respond to limitations of Soundex.
MD5	A message digest algorithm that produces a 128-bit hash value. This algorithm is commonly used to check data integrity.
Metaphone	Returns a Metaphone coded key of selected fields. Metaphone is an algorithm for coding words using their English pronunciation.
SpanishMetaphone	Returns a Metaphone coded key of selected fields for the Spanish language. This metaphone algorithm codes words using their Spanish pronunciation.

Metaphone 3 Improves upon the Metaphone and Double Metaphone algorithms with more exact consonant and internal vowel settings that allow you to produce words or names more or less closely matched to search terms on a phonetic basis. Metaphone 3 increases the accuracy of phonetic encoding to 98%. This option was developed to respond to limitations of Soundex.

3. In the **Field name** field, choose the field to which you want to apply the algorithm. For example, if you chose the soundex algorithm and chose a field named City, the ID would be generated by applying the soundex algorithm to the data in the City field.
4. If you selected the substring algorithm, specify the portion of the field you want to use in the substring:
 - a) In the **Start position** field, specify the position in the field where you want the substring to begin.
 - b) In the **Length** field, select the number of characters from the start position that you want to include in the substring.

For example, say you have this data in a field named LastName:

Augustine

If you specified 3 as the start position and 6 as the end position, the substring would produce:

gustin

5. Check the **Remove noise characters** box to remove all non-numeric and non-alpha characters such as hyphens, white space, and other special characters from the field before applying the algorithm.
6. For consonant and substring algorithms, you can sort the data in the field before applying the algorithm by checking the **Sort input** box. You can then choose to sort either the characters in the field or terms in the field in alphabetical order.
7. Click **OK** to save your settings.
8. Repeat as needed if you want to add additional algorithms to produce a more complex ID.

Note: The unique key definition is always displayed in a different color and cannot be deleted.

Defining a Non-Unique ID

Unique ID Generator can be used to generate a non-unique key using one of the key generation algorithms. In non-unique mode, you can create keys to use for matching. This may be useful in a data warehouse where you have already added keys to a dimension and you want to generate a key for new records in order to see if the new records match an existing record.

1. In the Unique ID Generator stage, on the **Rules** tab, click **Modify**.
2. Select **Off**.

This turns off the unique ID portion of the ID generation rules. With this option off, only the algorithm you choose in the steps below are used to create the ID. This means that any records that have the same data in the fields you use to generate the ID will have the same ID. You can then use the ID for matching.

3. Click **OK**.
4. At the warning prompt, click **Yes**.
5. In the Unique ID Generator stage, click **Add**.
6. In the **Algorithm** field, select the algorithm you want to use to generate additional information in the ID.

Consonant	Returns specified fields with consonants removed.
Double Metaphone	Returns a code based on a phonetic representation of their characters. Double Metaphone is an improved version of the Metaphone algorithm, and attempts to account for the many irregularities found in different languages.
Koeln	Indexes names by sound as they are pronounced in German. Allows names with the same pronunciation to be encoded to the same representation so that they can be matched, despite minor differences in spelling. The result is always a sequence of numbers; special characters and white spaces are ignored. This option was developed to respond to limitations of Soundex.
MD5	A message digest algorithm that produces a 128-bit hash value. This algorithm is commonly used to check data integrity.
Metaphone	Returns a Metaphone coded key of selected fields. Metaphone is an algorithm for coding words using their English pronunciation.
SpanishMetaphone	Returns a Metaphone coded key of selected fields for the Spanish language. This metaphone algorithm codes words using their Spanish pronunciation.
Metaphone 3	Improves upon the Metaphone and Double Metaphone algorithms with more exact consonant and internal vowel settings that allow you to produce words or names more or less closely matched to search terms on a phonetic basis. Metaphone 3 increases the accuracy of phonetic encoding to 98%. This option was developed to respond to limitations of Soundex.

7. In the **Field name** field, choose the field to which you want to apply the algorithm. For example, if you chose the soundex algorithm and chose a field named City, the ID would be generated by applying the soundex algorithm to the data in the City field.
8. If you selected the substring algorithm, specify the portion of the field you want to use in the substring:
 - a) In the **Start position** field, specify the position in the field where you want the substring to begin.

- b) In the **Length** field, select the number of characters from the start position that you want to include in the substring.

For example, say you have this data in a field named LastName:

Augustine

If you specified 3 as the start position and 6 as the end position, the substring would produce:

gustin

9. Check the **Remove noise characters** box to remove all non-numeric and non-alpha characters such as hyphens, white space, and other special characters from the field before applying the algorithm.
10. For consonant and substring algorithms, you can sort the data in the field before applying the algorithm by checking the **Sort input** box. You can then choose to sort either the characters in the field or terms in the field in alphabetical order.
11. Click **OK** to save your settings.
12. Repeat as needed if you want to add additional algorithms to produce a more complex ID.

Note: The unique key definition is always displayed in a different color and cannot be deleted.

Module Stages

Advanced Matching stages

Best of Breed

Best of Breed consolidates duplicate records by selecting the best data in a duplicate record collection and creating a new consolidated record using the best data. This "super" record is known as the best of breed record. You define the rules to use in selecting records to process. When processing completes, the best of breed record is retained by the system.

Options

The following table lists the options for Best of Breed.

Option Name	Description / Valid Values
Group by	Specifies the field to use to create groups of records to merge into a single best of breed record, creating one best of breed record from each group. In cases where you have used a matching stage earlier in the dataflow, you should select the CollectionNumber field to use the collections created by the matching stage as the groups. However, if you want to group records by some other field, choose the field here. For example, if you want to merge all records that have the same value in the AccountNumber field into one best of breed record, you would select AccountNumber.
Sort	If you specify a field in the Group by field, check this box to sort the records by the value in the field you chose. This option is enabled by default.

Option Name	Description / Valid Values
Advanced	<p>Click this button to specify sort performance options. By default, the sort performance options specified in Management Console, which are the default performance options for your system, are in effect. If you want to override your system's default performance options, check the Override sort performance options box then specify the values you want in these fields:</p> <p>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.</p> <p style="text-align: right;">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.</p> <p>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:</p> $\frac{(NumberOfRecords \times 2)}{InMemoryRecordLimit} = NumberOfTempFilesN$ <p style="text-align: right;">Note: The maximum number of temporary files cannot be more than 1,000.</p> <p>Enable compression Specifies that temporary files are compressed when they are written to disk.</p> <p style="text-align: right;">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: $(InMemoryRecordLimit \times MaxNumberOfTempFiles \div 2) \geq TotalNumberOfRecords$</p>
Keep original records	Select this option to retain all records in the collection along with the best of breed record. Clear the option if you want only the best of breed record.
Use first record	Select this option if you want Best of Breed to automatically select the first record in the collection as the template record. The template record is the record upon which the best of breed record is based.

Option Name	Description / Valid Values
Define template record	Select this option to define rules for selecting the template record. For more information, see Defining Template Record Rules on page 99.

Defining Template Record Rules

In Best of Breed processing, the template record is the record in a collection that is used to create the best of breed record. The template record is used as the starting point for constructing the best of breed record and is modified based on the best of breed settings you define. The Best of Breed stage can select the template record automatically, or you can define rules for selecting the template record. This topic describes how to define rules for selecting the template record.

Template rules are written by specifying the field name, an operator, a value type, and a value. Here is an example of template record options:

Field Name: MatchScore
 Field Type: Numeric
 Operator: Equal
 Value Type: String
 Value: 100

This template rule selects the record in the collection where the Match Score is equal to the value of 100.

The following procedure describes how to define a template record rule in the Best of Breed stage.

1. In the Best of Breed stage, under **Template Record Settings**, select the option **Define template record**.
2. In the tree, click **Rules**.
3. Click **Add Rule**.
4. Complete the following fields.

Option	Description				
Field name	Specifies the name of the dataflow field whose value you want to evaluate to determine if the record should be the template record.				
Field Type	Specifies the type of data in the field. One of the following: <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 10px;">Non-Numeric</td> <td>Choose this option if the field contains non-numeric data (for example, string data).</td> </tr> <tr> <td>Numeric</td> <td>Choose this option if the field contains numeric data (for example, double, float, and so on).</td> </tr> </table>	Non-Numeric	Choose this option if the field contains non-numeric data (for example, string data).	Numeric	Choose this option if the field contains numeric data (for example, double, float, and so on).
Non-Numeric	Choose this option if the field contains non-numeric data (for example, string data).				
Numeric	Choose this option if the field contains numeric data (for example, double, float, and so on).				

Option	Description
Operator	Specifies the type of comparison you want to use to evaluate the field. One of the following:
Contains	Determines if the field contains the value specified. For example, "sailboat" contains the value "boat".
Equal	Determines if the field contains the exact value specified.
Greater Than	Determines if the field value is greater than the value specified. This operation only works on numeric fields.
Greater Than Or Equal To	Determines if the field value is greater than or equal to the value specified. This operation only works on numeric fields.
Highest	Compares the field's value for all the records group and determines which record has the highest value in the field. For example, if the fields in the group contain values of 10, 20, 30, and 100, the record with the field value 100 would be selected. This operation only works on numeric fields. If multiple records are tied for the longest value, one record is selected.
Is Empty	Determines if the field contains no value.
Is Not Empty	Determines if the field contains any value.
Less Than	Determines if the field value is less than the value specified. This operation only works on numeric fields.
Less Than Or Equal To	Determines if the field value is less than or equal to the value specified. This operation only works on numeric fields.
Longest	Compares the field's value for all the records group and determines which record has the longest (in bytes) value in the field. For example, if the group contains the values "Mike" and "Michael", the record with the value "Michael" would be selected. If multiple records are tied for the longest value, one record is selected.
Lowest	Compares the field's value for all the records group and determines which record has the lowest value in the field. For example, if the fields in the group contain values of 10, 20, 30, and 100, the record with the field value 10 would be selected. This operation only works on numeric fields. If multiple records are tied for the longest value, one record is selected.
Most Common	Determines if the field value contains the value that occurs most frequently in this field among the records in the group. If two or more values are most common, no action is taken.
Not Equal	Determines if the field value is not the same as the value specified.

Option	Description
Value type	<p>Specifies the type of value you want to compare to the field's value. One of the following:</p> <p>Note: This option is not available if you select the operator Highest, Lowest, or Longest.</p> <p>Field Choose this option if you want to compare another dataflow field's value to the field.</p> <p>String Choose this option if you want to compare the field to a specific value.</p>
Value	<p>Specifies the value to compare to the field's value. If you selected Field in the Field type field, select a dataflow field. If you selected String in the Value type field, type the value you want to use in the comparison.</p> <p>Note: This option is not available if you select the operator Highest, Lowest, or Longest.</p>

- Click **OK**.
- If you want to specify additional rules, click **Add Rule**.

If you add additional rules, you will have to select a logical operator to use between each rule. Choose **And** if you want the new rule and the previous rule to both pass in order for it to be selected as the template record. Select **Or** if you want either the previous rule or the new rule to pass in order for the record to be selected as the template record.

You have now configured rules to use to select the template record. Configure the best of breed settings to complete the configuration of the Best of Breed stage.

Defining Best of Breed Rules and Actions

Best of Breed rules and actions work together to determine which fields from duplicate records in a collection to copy to the Best of Breed record. Rules test values in a record and if the record passes the rules, the data is copied from the record to the template record. Actions define which data to copy, and which field in the template record should receive the data. After all the rules and actions are executed, the template record will be the best of breed record.

Rules and actions can be grouped together into conditions, and you can have multiple conditions. This allows you

- In the Best of Breed stage, under **Best of Breed Settings**, click the **Rules** node in the tree.
- Click **Add Rule**.
- Complete the following fields:

Option	Description
Field name	Specifies the name of the dataflow field whose value you want to evaluate to determine if the condition is met and the associated actions should be taken.
Field Type	Specifies the type of data in the field. One of the following: Non-Numeric Choose this option if the field contains non-numeric data (for example, string data). Numeric Choose this option if the field contains numeric data (for example, double, float, and so on).

Option	Description
Operator	Specifies the type of comparison you want to use to evaluate the field. One of the following:
Contains	Determines if the field contains the value specified. For example, "sailboat" contains the value "boat".
Equal	Determines if the field contains the exact value specified.
Greater Than	Determines if the field value is greater than the value specified. This operation only works on numeric fields.
Greater Than Or Equal To	Determines if the field value is greater than or equal to the value specified. This operation only works on numeric fields.
Highest	Compares the field's value for all the records group and determines which record has the highest value in the field. For example, if the fields in the group contain values of 10, 20, 30, and 100, the record with the field value 100 would be selected. This operation only works on numeric fields. If multiple records are tied for the longest value, one record is selected.
Is Empty	Determines if the field contains no value.
Is Not Empty	Determines if the field contains any value.
Less Than	Determines if the field value is less than the value specified. This operation only works on numeric fields.
Less Than Or Equal To	Determines if the field value is less than or equal to the value specified. This operation only works on numeric fields.
Longest	Compares the field's value for all the records group and determines which record has the longest (in bytes) value in the field. For example, if the group contains the values "Mike" and "Michael", the record with the value "Michael" would be selected. If multiple records are tied for the longest value, one record is selected.
Lowest	Compares the field's value for all the records group and determines which record has the lowest value in the field. For example, if the fields in the group contain values of 10, 20, 30, and 100, the record with the field value 10 would be selected. This operation only works on numeric fields. If multiple records are tied for the longest value, one record is selected.
Most Common	Determines if the field value contains the value that occurs most frequently in this field among the records in the group. If two or more values are most common, no action is taken.
Not Equal	Determines if the field value is not the same as the value specified.

Option	Description
Value type	<p>Specifies the type of value you want to compare to the field's value. One of the following:</p> <p>Note: This option is not available if you select the operator Highest, Lowest, or Longest.</p> <p>Field Choose this option if you want to compare another dataflow field's value to the field.</p> <p>String Choose this option if you want to compare the field to a specific value.</p>
Value	<p>Specifies the value to compare to the field's value. If you selected Field in the Field type field, select a dataflow field. If you selected String in the Value type field, type the value you want to use in the comparison.</p> <p>Note: This option is not available if you select the operator Highest, Lowest, or Longest.</p>

- Click **OK**.
- If you want to specify additional rules for this condition, click **Add Rule**.

If you add additional rules, you will have to select a logical operator to use between each rule. Choose **And** if you want the new rule and the previous rule to both pass in order for the condition to be met and the associated actions taken. Select **Or** if you want either the previous rule or the new rule to pass in order for the condition to be met.

- Click the **Actions** node in the tree.
- Click **Add Action**.
- Complete the following fields.

Option	Description
Source type	<p>Specifies the type of data to copy to the best of breed record. One of the following.</p> <p>Field Choose this option if you want to copy a value from a field to the best of breed record.</p> <p>String Choose this option if you want to copy a constant value to the best of breed record.</p>
Source data	<p>Specifies the data to copy to the best of breed record. If the source type is Field, select the field whose value you want to copy to the destination field. If the source type is String, specify a constant value to copy to the destination field.</p>

Option	Description
Destination	Specifies the field in the best of breed record to which you want to copy the data specified in the Source data field.
Accumulate source data	<p>If the data in the Source data field is numeric data, you can enable this option to combine the source data for all duplicate records and put the total value in the best of breed record.</p> <p>For example, if there were three duplicate records in the group and they contained these values in the Deposits field:</p> <p>100.00 20.00 5.00</p> <p>Then all three values would be combined and the total value, 125.00, would be put in the best of breed record's Deposits field.</p>

9. Click **OK**.
10. If you want to specify additional actions to take for this condition, click **Add Action** and repeat the above steps.
11. To add another condition, click the root condition in the tree then click **Add Condition**.

Example Best of Breed Rule and Action

This Best of Breed rule selects the record where the Match Score is equal to the value of 100. The Account Number data that corresponds to the selected field is then copied to the AccountNumber field on the Best of Breed record.

Rule

Field Name: MatchScore
Field Type: Numeric
Operator: Equal
Value Type: String
Value: 100

Action

Source Type: Field
Source Data: AccountNumber
Destination: AccountNumber

Output

Table 9: Best of Breed Output

Field Name	Format	Description / Valid Values
CollectionRecordType	String	Identifies the template and Best of Breed records in a collection of duplicate records. The possible values are: <ul style="list-style-type: none"> Primary The record is the selected template record in a collection. Secondary The record is not the selected template record in a collection. BestOfBreed The record is the newly created best of breed record in the collection. <p>Note: The Primary and Secondary values are generated only when you select the Define template record option in the Best of Breed Options window.</p>

Candidate Finder

Candidate Finder obtains the candidate records that will form the set of potential matches. Database searches work in conjunction with Transactional Match, and Search Index searches work independently from Transactional Match. Depending on the format of your data, Candidate Finder may also need to parse the name or address of the suspect record, the candidate records, or both.

Candidate Finder also enables full-text index searches and helps in defining both simple and complex search criteria against characters and text using various search types (Any Word Starts With, Contains, Contains All, Contains Any, Contains None, Fuzzy, Pattern, Proximity, Range, Wildcard) and conditions (All True, Any True, None True).

Database Options

The Candidate Finder dialog enables you to define SQL statements that retrieve potential match candidates from a database, as well as map the columns that you select from the database to the field names that are defined in your dataflow.

Table 10: Candidate Finder Database Options

Option Name	Description / Valid Values
Finder type	Select Database.
Connection	Select the database that contains the candidate records. You can select any connection configured in Management Console. To connect to a database not listed, configure a connection to that database in Management Console, then close and reopen Candidate Finder to refresh the connection list. Note: The Dataflow Options feature in Enterprise Designer enables the connection name to be exposed for configuration at runtime.
SQL statement	Type a SQL statement in the text box as described in Defining the SQL Query on page 107
Field Map tab	Choose field mapping settings as described in Mapping Database Columns to Stage Fields on page 108.
Preview tab	Click this tab to enter a sample match key to test your SQL <code>SELECT</code> statement or your index query.

Defining the SQL Query

You can type any valid SQL select statement into the text box on the **Candidate Finder Options** dialog.

Note: `Select *` is not valid.

For example, assume you have a table in your database called `Customer_Table` that has the following columns:

- `Customer_Table`
- `Cust_Name`
- `Cust_Address`
- `Cust_City`
- `Cust_State`
- `Cust_Zip`

To retrieve all the rows from the database, you might construct a query similar to the following:

```
SELECT Cust_Name, Cust_Address, Cust_City, Cust_State, Cust_Zip from
Customer_Table;
```

You will rarely want to match your transaction against all the rows in the database. To return only relevant candidate records, add a `WHERE` clause using variable substitution. Variable substitution refers to a special notation that you will use to cause the Candidate Selection engine to replace the variable with the actual data from your suspect record.

To use variable substitution, enclose the field name in braces preceded by a dollar sign using the form `${FieldName}`. For example, the following query will return only those records that have a value in `Cust_Zip` that matches the value in `PostalCode` on the suspect record.

```
SELECT Cust_Name, Cust_Address, Cust_City, Cust_State, Cust_Zip
FROM Customer_Table
WHERE Cust_Zip = ${PostalCode};
```

For SQL 2000, the data type needs to be identical to the data type for Candidate Finder. The JDBC driver sets the Candidate Finder input variable (Ex: `${MatchKey}`) that is used in the `WHERE` clause to a data type of `nVarChar(4000)`. If the data in the database is set to a data type of `VarChar`, SQL Server will ignore the index on the database. If the index is ignored, then performance will be degraded. Therefore, use the following query for SQL 2000:

```
SELECT Cust_Name, Cust_Address, Cust_City, Cust_State, Cust_Zip
FROM Customer_Table
WHERE Cust_Zip = CAST(${PostalCode} AS VARCHAR(255));
```

Mapping Database Columns to Stage Fields

If the column names in your database match the Component Field names exactly, they are automatically mapped to the corresponding Stage Fields. If they are not named exactly the same, you will need to use the Selected Fields (columns from the database) to map to the Stage Fields (field names defined in the dataflow).

For example, consider a table named `Customer_Table` with the following columns:

- `Cust_Name`
- `Cust_Address`
- `Cust_City`
- `Cust_State`
- `Cust_Zip`

When you retrieve these records from the database, you need to map the column names to the field names that are used by Transactional Match and other components in your dataflow. For example, `Cust_Address` might be mapped to `AddressLine1`, and `Cust_Zip` would be mapped to `PostalCode`.

1. Select the drop-down list under **Selected Fields** in the **Candidate Finder Options** dialog. Then, select the database column `Cust_Zip`.

2. Select the drop-down list under **Stage Fields**. Then, select the field to which you want to map. For example, if you want to map Cust_Zip to Postal Code, first select Cust_Zip under Selected fields and then select PostalCode on the corresponding Stage Field row.

Alternate Method for Mapping Fields

You can use special notation in your SQL query to perform the mapping. To do this, enclose the field name you want to map to in braces after the column name in your query. When you do this, the selected fields are automatically mapped to the corresponding stage fields.

For example,

```
select Cust_Name {Name}, Cust_Address {AddressLine1},
       Cust_City {City}, Cust_State {StateProvince},
       Cust_Zip {PostalCode}
from Customer
where Cust_Zip = ${PostalCode};
```

Configuring the Connection Name at Runtime

The Connection name can be configured and passed at runtime if it is exposed as a dataflow option. This enables you to run your dataflow while using a different connection name.

1. In Enterprise Designer, open a dataflow that uses the Candidate Finder stage.
2. Save and expose that dataflow.
3. Go to **Edit > Dataflow Options**.
4. In the **Map dataflow options to stages** table, expand Candidate Finder and edit options as necessary. Check the box for the option you want to edit, then change the value in the **Default value** drop-down.
5. Optional: Change the name of the options in the **Option label** field.
6. Click **OK** twice.

Search Index Options

The Candidate Finder dialog enables you to conduct a simple search that matches input field values within search indexes or an advanced search to build matching rules that retrieve potential match candidates from search indexes.

Simple Search Index Options

Table 11: Candidate Finder Options

Option Name	Description / Valid Values
Finder type	Select Search Index.

Option Name	Description / Valid Values
Name	Select the appropriate index that was created using the Write to Search Index on page 152 stage under the Advanced Matching deployed stages in Enterprise Designer.
Starting record	Enter the record number on which search results should begin. The default is 1.
Maximum results	Enter the maximum number of results you want the search index to return. Default is 10. Note: If the maximum results is arbitrarily large, process those in batches, using the Fetch Batch Size field.
Fetch Batch Size	If the Maximum results is arbitrarily large, enter the size of batches in which you want the results to be processed. This optimizes processing of large number of records. Default is 10000. The recommended Fetch Batch Size is a value lesser than Maximum results and if the Fetch Batch Size is greater than Maximum results , the records are processed in a single batch. Note: This field is applicable only to cluster supported search engine and not to the legacy search engine.
Return match count	Returns the total number of matches that were made. For example, if you use the default of "10" for the Maximum results field above, only 10 results will be returned. However, if you check this box, the TotalMatchCount output field will tell you how many matches were made during processing.
Index search type	Determines the type of index search you want to conduct. Select Simple search .
Index Fields	Select the index field(s) you want to use for comparison in the simple search.
Input field	Select the input field you want to use for comparison in the simple search.

Option Name	Description / Valid Values
-------------	----------------------------

Input analyzer	
----------------	--

Option Name	Description / Valid Values
-------------	----------------------------

Specify which analyzer to use to tokenize the input string. One of these:

- **Standard**—Provides a grammar-based tokenizer that contains a superset of the Whitespace and Stop Word analyzers. Understands English punctuation for breaking down words, knows words to ignore (via the Stop Word Analyzer), and performs technically case-insensitive searching by conducting lowercase comparisons. For example, the string “Precisely Software” would be returned as two tokens: “Precisely” and “Software”.
- **Whitespace**—Separates tokens with whitespace. Somewhat of a subset of the Standard Analyzer in that it understands word breaks in English text based on spaces and line breaks.
- **StopWord**—Removes articles such as “the,” “and,” and “a” to shrink the index size and increase performance.
- **Keyword**—Creates a single token from a stream of data. For example, the string “Precisely Software” would be returned as just one token “Precisely Software”.
- **Russian**—Supports Russian-language indexes and type-ahead services. Also supports many stop words and removes articles such as “and,” “I,” and “you” to shrink the index size and increase performance.
- **German**—Supports German-language indexes and type-ahead services. Also supports many stop words and removes articles such as “the,” “and,” and “a” to shrink the index size and increase performance.
- **Danish**—Supports Danish-language indexes and type-ahead services. Also supports many stop words and removes articles such as “at,” “and,” and “a” to shrink the index size and increase performance.
- **Dutch**—Supports Dutch-language indexes and type-ahead services. Also supports many stop words and removes articles such as “the,” “and,” and “a” to shrink the index size and increase performance.
- **Finnish**—Supports Finnish-language indexes and type-ahead services. Also supports many stop words and removes articles such as “is,” “and,” and “of” to shrink the index size and increase performance.
- **French**—Supports French-language indexes and type-ahead services. Also supports many stop words and removes articles such as “the,” “and,” and “a” to shrink the index size and increase performance.
- **Hungarian**—Supports Hungarian-language indexes and type-ahead services. Also supports many stop words and removes articles such as “the,” “and,” and “a” to shrink the index size and increase performance.
- **Italian**—Supports Italian-language indexes and type-ahead services. Also supports many stop words and removes articles such as “the,” “and,” and “a” to shrink the index size and increase performance.
- **Norwegian**—Supports Norwegian-language indexes and type-ahead services. Also supports many stop words and removes articles such as “the,” “and,” and “a” to shrink the index size and increase performance.
- **Portuguese**—Supports Portuguese-language indexes and type-ahead services. Also supports many stop words and removes articles such as “the,” “and,” and “a” to shrink the index size and increase performance.
- **Spanish**—Supports Spanish-language indexes and type-ahead services. Also supports many stop words and removes articles such as “the,” “and,” and “a” to shrink the index size and increase performance.
- **Swedish**—Supports Swedish-language indexes and type-ahead services. Also

Option Name	Description / Valid Values
	<p>supports many stop words and removes articles such as "the" "and," and "a" to shrink the index size and increase performance.</p> <ul style="list-style-type: none"> • Hindi—Supports Hindi-language indexes and type-ahead services. Also supports many stop words and removes articles such as "by" "and," and "a" to shrink the index size and increase performance.
Output Fields tab	<p>Check the Include box to select which stored fields should be included in the output.</p> <p>Note: If the input field is from an earlier stage in the dataflow and it has the same name as the store field name from the search index, the values from the input field will overwrite the values in the output field.</p>

Advanced Search Index Options

Table 12: Candidate Finder Options

Option Name	Description / Valid Values
Finder type	Select Search Index.
Name	Select the appropriate index that was created using the Write to Search Index on page 152 stage under the Advanced Matching deployed stages in Enterprise Designer.
Starting record	Enter the record number on which search results should begin. The default is 1.
Maximum results	<p>Enter the maximum number of responses you want the index search to return. The default is 10.</p> <p>Note: If the maximum results is arbitrarily large, process those in batches, using the Fetch Batch Size field.</p>

Option Name	Description / Valid Values
Fetch Batch Size	<p>If the Maximum results is arbitrarily large, enter the size of batches in which you want the results to be processed. This optimizes processing of large number of records. Default is 10000.</p> <p>The recommended Fetch Batch Size is a value lesser than Maximum results and if the Fetch Batch Size is greater than Maximum results, the records are processed in a single batch.</p> <p>Note: This field is applicable only to cluster supported search engine and not to the legacy search engine.</p>
Sort	<p>Sorts the candidate records on the basis of indexed fields while running a search query.</p> <p>Select the Sort check-box, the desired index field from the Sort by drop-down list, and select Ascending or Descending from the Order by drop-down list.</p> <p>Note: You can perform sorting only on String Fields with Keyword Analyzer and Numeric fields.</p>
Return match count	<p>Returns the total number of matches that were made. For example, if you use the default of "10" for the Maximum results field above, only 10 results will be returned. However, if you check this box, the TotalMatchCount output field will tell you how many matches were made during processing.</p>
Relevance	<p>Controls the relevance of the Index Field.</p>
Index search type	<p>Determines the type of index search you want to conduct. Select Advanced search.</p>
Add Parent button	<p>Access Parent Options.</p>
Parent options—Name	<p>Enter a name for the parent.</p>

Option Name	Description / Valid Values
Parent options—Searching method	<p>Specify how to determine if a parent is a match or a non-match. One of these:</p> <p>All true—A parent is considered a match if all children are determined to match. This method creates an "AND" connector between children.</p> <p>Any true—A parent is considered a match if at least one child is determined to match. This method creates an "OR" connector between children.</p> <p>None true—A parent is considered a match if none of the children is determined to match. This method creates a "NOT" connector between children.</p>
Add Child button	Access Child Options.
Child options—Index field	Select the index field you want to use for comparison in the advanced search.
Child options—Search type	Specifies the searching/matching criteria that determines whether the input data is searched/matched with the indexed data. All searches are case insensitive.
Child options—Input field	Select the input field you want to use for comparison in the advanced search.
Any Word/Phrase Starts With	<p>Determines whether the text contained in the search index field begins with the text that is contained in the input field.</p> <p>For example, text in the input field "tech" would be considered a match for search index fields containing "Technical", "Technology", "Technologies", "Technician" or even "National University of Technical Sciences". Likewise, a phrase in the input field "DEF Sof" would be considered a match for search index fields containing "ABC DEF Software", "DEF Software", and "DEF Software India" but it would not be a match for search index fields containing "Software DEF" or "DEF ABC Software".</p>
Contains	<p>Determines whether the search index field contains the data from the input field. This search type considers the sequence of words in the input field while searching the search index field. For example, input field data "Precisely" and "Precisely Software" would be contained in a search index field of "Precisely Software Inc."</p>
Contains All	<p>Determines whether all alphanumeric words from the input field are contained in the search index field. This search type does not consider the sequence of words in the input field while searching the search index field.</p>
Contains Any	<p>Determines whether any of the alphanumeric words from the input field is contained in the search index field.</p>

Option Name	Description / Valid Values
Contains None	Determines whether none of the alphanumeric words from the input field is contained in the search index field.
Fuzzy	<p>Determines the similarity between two alphanumeric words based on the number of deletions, insertions, or substitutions required to transform one word into another.</p> <p>Use the Maximum edits parameter to set a limit on the number of edits allowed to be considered a successful match:</p> <ul style="list-style-type: none"> • 0—Allows for no deletions, insertions, or substitutions. The input field data and the search index field data must be identical. • 1—Allows for no more than one deletion, insertion, or substitution. For example, an input field containing "Barton" will match a search index field containing "Carton". • 2—Allows for no more than two deletions, insertions, or substitutions. For example, an input field containing "Barton" will match a search index field containing "Martin". <p>The Fuzzy search type is used for single-word searches only. Click Ignore extra words to have Candidate Finder consider only the first word in the field when comparing the input field to the index field. For example, if the index field says "Xyz" and the input field says "Xyz Abc", they would not be considered a match because of "Abc". However, if you check this box, "Abc" would be ignored and with "Xyz" being the first word, the two words would be considered a match.</p>
Numeric	<p>Determines whether numbers from the input field are contained in the search index field.</p> <p>The Numeric search type is used for single-word searches only.</p> <p>Click Ignore extra words to have Candidate Finder consider only the first word in the field when comparing the input field to the index field.</p>
Pattern	<p>Determines whether the text pattern of the input field matches the text pattern of the search criteria. You can further refine the text pattern in the Pattern string field. For example, if the input field contains "nlm" and the pattern defined is "a*b?c" then it will match the following words "Neelam", "nelam", "neelum", "nilam", and so on.</p> <p>The Pattern search type is used for single-word searches only. Click Ignore extra words to have Candidate Finder consider only the first word in the field when comparing the input field to the index field.</p>

Option Name	Description / Valid Values
Proximity	<p>Determines whether words in the input fields are within a certain distance of each other.</p> <ul style="list-style-type: none"> • Define the input First input field and Second input field you want to search for in the index. • Use the Distance parameter to determine the maximum allowed distance between the words specified in the First field and Second field in order to be considered a match. <p>For example, you could successfully use this search type to look for First field "Spectrum" and Second field "Precisely" within ten words of each other in a search index field containing the sentence "Spectrum Technology Platform is a product of Precisely Software Inc."</p> <p>The Proximity search type is used for single-word searches only. Click Ignore extra words to have Candidate Finder consider only the first word in the field when comparing the input field to the index field.</p>
Range	<p>Performs an inclusive searches for terms within a range, which is specified using a Lower bound field (starting term) and an Upper bound field (ending term). All alphanumeric words are arranged lexicographically in the search index field.</p> <ul style="list-style-type: none"> • Use the Lower bound field parameter to select the field to be used as the starting term. • Use the Upper bound field parameter to select the field to be used as the ending term. <p>For example, if you searched postal codes from 20001 (defined in the Lower bound field) to 20009 (defined in the Upper bound field), the search would return all addresses with postal codes within that range.</p> <p>The Range search type is used for single-word searches only. Click Ignore extra words to have Candidate Finder consider only the first word in the field when comparing the input field to the index field.</p>
Wildcard	<p>Searches using single or multiple Wildcard characters.</p> <p>Select the Position in your input file where you are inserting the wildcard character.</p> <p>The Wildcard search type is used for single-word searches only. Click Ignore extra words to have Candidate Finder consider only the first word in the field when comparing the input field to the index field.</p>

Option Name	Description / Valid Values
Child options—Relevance factor	<p>Control the relevance of a child field by entering any positive number up to 100 here. The number can be less than "1" also; for instance, ".05" would be valid.</p> <p>The higher the boost factor, the more relevant the field will be. For example, if you want results from the Firm Name field to be more relevant than the results from other fields, select "Firm Name" from the Index field name and enter "5" here.</p> <p>Note: By default, this option is disabled. Select the check box to enable it.</p>
Ignore Blanks	<p>Clear this check-box if you want the query to take into account the blank input file fields.</p> <p>Note: By default the query ignores the blank fields.</p>
Output Fields tab	<p>Check the Include box to select which stored fields should be included in the output.</p> <p>Note: If the input field is from an earlier stage in the dataflow and it has the same name as the store field name from the search index, the values from the input field will overwrite the values in the output field.</p>

Configuring Options at Runtime

Some Candidate Finder options can be configured and passed at runtime if they are exposed as dataflow options. This enables you to run your dataflow while using different configurations. These are the available dataflow options for Candidate Finder:

- **ConnectionName**—The name of the database that contains the candidate records.
- **SearchIndexName**—The name of the search index used in the Candidate Finder dataflow.
- **StartingRecord**—The record number on which search results should begin.
- **MaximumResults**—The maximum number of responses you want the index search to return.
- **ReturnMatchCount**—The total number of matches that were made. This field is useful if you enter a lower number in the MaximumResults field but want to know the total number of matches that were made.

To define Candidate Finder options at runtime:

1. In Enterprise Designer, open a dataflow that uses the Candidate Finder stage.
2. Save and expose that dataflow.
3. Go to **EditDataflow Options**.
4. In the **Map dataflow options to stages** table, expand Candidate Finder and edit options as necessary. Check the box for the option you want to edit, then change the value in the **Default value** drop-down.

5. Optional: Change the name of the options in the **Option label** field.
6. Click **OK** twice.

Output

Table 13: Candidate Finder Outputs

Field Name	Format	Description / Valid Values
CandidateCount	String	This field indicates the total number of candidates returned during processing.
CandidateGroup	String	This field identifies a grouping of a suspect record and its candidates. Each suspect record is given a CandidateGroup number. The candidates for that suspect are given the same CandidateGroup number. For example, if John Smith is a suspect record and its candidate records are John Smith and Jon Smth, then all three records would have the same CandidateGroup value.
HasDuplicates	String	Identifies whether candidates are detected or not. The possible values are: Y - The record is suspect and has candidates N - The record is suspect and doesn't have candidates D - The record is a candidate record
TotalMatchCount	String	This field indicates the total number of matches made during the processing.
TransactionRecordType	String	One of the following: Suspect A suspect record is used as input to a query. Candidate A candidate record is a result returned from a query.

Duplicate Synchronization

Duplicate Synchronization determines which fields from a collection of records to copy to the corresponding fields of all records in the collection. You can specify the rules that records must satisfy in order to copy the field data to the other records in the collection. When processing has been completed, all records in the collection are retained.

Options

The table lists the options for the **Duplicate Synchronization** stage.

Option Name	Description / Valid Values
Group by	Specifies the field to use to create groups of records to synchronize. In cases where you have used a matching stage earlier in the dataflow, such as Interflow Match, Intraflow Match, or Transactional Match, you should select the CollectionNumber field to use the collections created by the matching stage as the groups. However, if you want to group records by some other field, choose the field here. For example, if you want to synchronize records that have the same value in the AccountNumber field, you would select AccountNumber.
Sort	If you specify a field in the Group by field, check this box to sort the records by the value in the field you chose. This option is enabled by default.

Option Name	Description / Valid Values
Advanced	<p data-bbox="548 380 1425 527">Click this button to specify sort performance options. By default, the sort performance options specified in Management Console, which are the default performance options for your system, are in effect. If you want to override your system's default performance options, check the Override sort performance options box then specify the values you want in these fields:</p> <p data-bbox="557 541 1425 793">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.</p> <p data-bbox="813 810 1425 926">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.</p> <p data-bbox="557 968 1425 1220">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:</p> $\frac{(\text{NumberOfRecords} \times 2)}{\text{InMemoryRecordLimit}} = \text{NumberOfTempFiles}$ <p data-bbox="813 1304 1425 1356">Note: The maximum number of temporary files cannot be more than 1,000.</p> <p data-bbox="557 1398 1425 1451">Enable compression Specifies that temporary files are compressed when they are written to disk.</p> <p data-bbox="634 1486 1425 1602">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: $(\text{InMemoryRecordLimit} \times \text{MaxNumberOfTempFiles} \div 2) \geq \text{TotalNumberOfRecords}$</p>

Rules

Duplicate Synchronization rules determine which records should have their data copied to all other records in the collection.

To add a rule, select Rules in the rule hierarchy and click **Add Rule**

If you specify multiple rules, you will have to select a logical operator to use between each rule. Choose **And** if you want the new rule and the previous rule to both pass in order for the condition to be met. Select **Or** if you want either the previous rule or the new rule to pass in order for the condition to be met.

Option	Description
Field name	Specifies the name of the dataflow field whose value you want to evaluate to determine whether to filter the record.
Field Type	Specifies the type of data in the field. One of the following: Non-Numeric Choose this option if the field contains non-numeric data (for example, string data). Numeric Choose this option if the field contains numeric data (for example, double, float, and so on).

Option	Description
Operator	Specifies the type of comparison you want to use to evaluate the field. One of the following:
Contains	Determines if the field contains the value specified. For example, "sailboat" contains the value "boat".
Equal	Determines if the field contains the exact value specified.
Greater Than	Determines if the field value is greater than the value specified. This operation only works on numeric fields.
Greater Than Or Equal To	Determines if the field value is greater than or equal to the value specified. This operation only works on numeric fields.
Highest	Compares the field's value for all the records group and determines which record has the highest value in the field. For example, if the fields in the group contain values of 10, 20, 30, and 100, the record with the field value 100 would be selected. This operation only works on numeric fields. If multiple records are tied for the longest value, one record is selected.
Is Empty	Determines if the field contains no value.
Is Not Empty	Determines if the field contains any value.
Less Than	Determines if the field value is less than the value specified. This operation only works on numeric fields.
Less Than Or Equal To	Determines if the field value is less than or equal to the value specified. This operation only works on numeric fields.
Longest	Compares the field's value for all the records group and determines which record has the longest (in bytes) value in the field. For example, if the group contains the values "Mike" and "Michael", the record with the value "Michael" would be selected. If multiple records are tied for the longest value, one record is selected.
Lowest	Compares the field's value for all the records group and determines which record has the lowest value in the field. For example, if the fields in the group contain values of 10, 20, 30, and 100, the record with the field value 10 would be selected. This operation only works on numeric fields. If multiple records are tied for the longest value, one record is selected.
Most Common	Determines if the field value contains the value that occurs most frequently in this field among the records in the group. If two or more values are most common, no action is taken.
Not Equal	Determines if the field value is not the same as the value specified.

Option	Description
Value type	<p>Specifies the type of value you want to compare to the field's value. One of the following:</p> <p>Note: This option is not available if you select the operator Highest, Lowest, or Longest.</p> <p>Field Choose this option if you want to compare another dataflow field's value to the field.</p> <p>String Choose this option if you want to compare the field to a specific value.</p>
Value	<p>Specifies the value to compare to the field's value. If you selected Field in the Field type field, select a dataflow field. If you selected String in the Value type field, type the value you want to use in the comparison.</p> <p>Note: This option is not available if you select the operator Highest, Lowest, or Longest.</p>

Actions

Actions determine which field to copy to other records in the group. To add an action, select Actions in the Duplicate Synchronization condition tree then click the **Add Action**. Use the following options to define the action.

Option	Description
Source type	<p>Specifies the type of data to copy to other records in the group. One of the following.</p> <p>Field Choose this option if you want to copy a value from a field to the other records in the group.</p> <p>String Choose this option if you want to copy a constant value to the other records in the group.</p>
Source data	<p>Specifies the data to copy to the other records in the group. If the source type is Field, select the field whose value you want to copy to the other records in the group. If the source type is String, specify a constant value to copy to the other records in the group.</p> <p>Note: In case the source data has null value it will not be copied to the other records of the group. The other records will rather retain their original values.</p>

Option	Description
Destination	Specifies the field in the other records to which you want to copy the data specified in the Source data field. For example, if you want to copy the data to the AccountBalance field in all the other records in the group, you would specify AccountBalance.

Example of a Duplicate Synchronization Rule and Action

This Duplicate Synchronization rule and action selects the record where the match score is 100 and copies the account number AccountNumber field in all the other records in the group.

Rule

Field Name: MatchScore
 Field Type: Numeric
 Operator: Equal
 Value Type: String
 Value: 100

Action

Source Type: Field
 Source Data: AccountNumber
 Destination: NewAccountNumber

Filter

The Filter stage retains or removes records from a group of records based on the rules you specify.

Options

The following table lists the options for the Filter stage.

Option Name	Description / Valid Values
Group by	Specifies the field to use to create groups of records to filter. The Filter stage will retain one or more records from each group, depending on how you configure the stage. In cases where you have used a matching stage earlier in the dataflow, such as Interflow Match, Intraflow Match, or Transactional Match, you should select the CollectionNumber field to use the collections created by the matching stage as the groups. However, if you want to group records by some other field, choose the field here. For example, if you want to filter out all but one record from records that have the same value in the AccountNumber field, you would select AccountNumber.

Option Name	Description / Valid Values
Sort	If you specify a field in the Group by field, check this box to sort the records by the value in the field you chose. This option is enabled by default.
Advanced	Click this button to specify sort performance options. By default, the sort performance options specified in Management Console, which are the default performance options for your system, are in effect. If you want to override your system's default performance options, check the Override sort performance options box then specify the values you want in these fields: <p data-bbox="558 680 691 737">In memory record limit</p> <p data-bbox="729 680 1417 926">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.</p> <p data-bbox="813 947 1409 1062">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.</p> <p data-bbox="558 1104 678 1220">Maximum number of temporary files</p> <p data-bbox="729 1104 1417 1350">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:</p> $\frac{(NumberOfRecords \times 2)}{InMemoryRecordLimit} = NumberOfTempFilesN$ <p data-bbox="813 1444 1398 1497">Note: The maximum number of temporary files cannot be more than 1,000.</p> <p data-bbox="558 1539 708 1596">Enable compression</p> <p data-bbox="729 1539 1417 1596">Specifies that temporary files are compressed when they are written to disk.</p> <p data-bbox="634 1623 1406 1738">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: $(InMemoryRecordLimit \times MaxNumberOfTempFiles \div 2) \geq TotalNumberOfRecords$</p>

Option Name	Description / Valid Values
Limit number of returned duplicate records	<p>Specifies the maximum number of records that are returned from each group. If you set this option to 1, you can define filter rules to determine which record in each group should be returned. If no rules are defined, the first record in each collection is returned and the rest are discarded. In this mode, the filter rules define which record will be retained.</p> <p>For example, if you define a rule where the record with the highest match score in a group is retained, and you set this option to 1, then the record with the highest match score in each group will survive and the other records in the group will be discarded.</p> <p>If you set this option to a value higher than one, you cannot specify filter rules.</p> <p>Note: In the event no records in the collection meet the defined rule criteria, then no records from the group are returned.</p>
Remove duplicates from collection	<p>Specifies to use filter rules to determine which records are removed from the collection. The remaining records in the collection are retained. When this option is selected, you must define a rule.</p> <p>Note: If a group contains only one record, the filter rules are ignored and the record is retained.</p>

Rule Options

Filter rules determine which records in a group to retain or remove. If you select the option **Limit number of returned duplicate records** then the rules determine which records survive the filter. If you select the option **Remove duplicates from collection** then the rules determine which records are removed from the dataflow.

To add a rule, select Rules in the rule hierarchy and click **Add Rule**

If you specify multiple rules, you will have to select a logical operator to use between each rule. Choose **And** if you want the new rule and the previous rule to both pass in order for the condition to be met. Select **Or** if you want either the previous rule or the new rule to pass in order for the condition to be met.

Note: You can only have one condition in a Filter stage. When you select Condition in the rule hierarchy, the buttons are grayed out.

Option	Description
Field name	Specifies the name of the dataflow field whose value you want to evaluate to determine whether to filter the record.
Field Type	Specifies the type of data in the field. One of the following: Non-Numeric Choose this option if the field contains non-numeric data (for example, string data). Numeric Choose this option if the field contains numeric data (for example, double, float, and so on).

Option	Description
Operator	Specifies the type of comparison you want to use to evaluate the field. One of the following:
Contains	Determines if the field contains the value specified. For example, "sailboat" contains the value "boat".
Equal	Determines if the field contains the exact value specified.
Greater Than	Determines if the field value is greater than the value specified. This operation only works on numeric fields.
Greater Than Or Equal To	Determines if the field value is greater than or equal to the value specified. This operation only works on numeric fields.
Highest	Compares the field's value for all the records group and determines which record has the highest value in the field. For example, if the fields in the group contain values of 10, 20, 30, and 100, the record with the field value 100 would be selected. This operation only works on numeric fields. If multiple records are tied for the longest value, one record is selected.
Is Empty	Determines if the field contains no value.
Is Not Empty	Determines if the field contains any value.
Less Than	Determines if the field value is less than the value specified. This operation only works on numeric fields.
Less Than Or Equal To	Determines if the field value is less than or equal to the value specified. This operation only works on numeric fields.
Longest	Compares the field's value for all the records group and determines which record has the longest (in bytes) value in the field. For example, if the group contains the values "Mike" and "Michael", the record with the value "Michael" would be selected. If multiple records are tied for the longest value, one record is selected.
Lowest	Compares the field's value for all the records group and determines which record has the lowest value in the field. For example, if the fields in the group contain values of 10, 20, 30, and 100, the record with the field value 10 would be selected. This operation only works on numeric fields. If multiple records are tied for the longest value, one record is selected.
Most Common	Determines if the field value contains the value that occurs most frequently in this field among the records in the group. If two or more values are most common, no action is taken.
Not Equal	Determines if the field value is not the same as the value specified.

Option	Description
Value type	<p>Specifies the type of value you want to compare to the field's value. One of the following:</p> <p>Note: This option is not available if you select the operator Highest, Lowest, or Longest.</p> <p>Field Choose this option if you want to compare another dataflow field's value to the field.</p> <p>String Choose this option if you want to compare the field to a specific value.</p>
Value	<p>Specifies the value to compare to the field's value. If you selected Field in the Field type field, select a dataflow field. If you selected String in the Value type field, type the value you want to use in the comparison.</p> <p>Note: This option is not available if you select the operator Highest, Lowest, or Longest.</p>

Example of a Filter Rule

This rule retains the record in each group with the highest value in the MatchScore field. Note that **Value** and **Value Type** options do not apply when the Operator is highest or lowest.

Field Name = MatchScore
 Field Type = Numeric
 Operator = Highest

This rule retains the record where the value in the AccountNumber is "12345".

Field Name = AccountNumber
 Field Type = Numeric
 Operator = Equals
 Value Type = String
 Value = 12345

Interflow Match

Interflow Match locates matches between similar data records across two input record streams. The first record stream is a source for suspect records and the second stream is a source for candidate records.

Using match group criteria (for example a match key), Interflow Match identifies a group of records that are potentially duplicates of a particular suspect record.

Each candidate is separately matched to the Suspect and is scored according to your match rules. If the candidate is a duplicate, it is assigned a collection number, the match record type is labeled a

duplicate, and written out; unmatched unique candidates may be written out at the user's option. When Interflow Match has exhausted all candidate records in the current match group, the matched suspect record is assigned a collection number that corresponds to its duplicate record. Or, if no matches were identified, the suspect is assigned a collection number of 0 and is labeled a unique record.

Note: Interflow Match only matches suspect records to candidate records. It does not attempt to match suspect records to other suspect records as is done in Intraflow Match.

The matching process for a particular suspect may terminate before matching all possible candidates if you have set a limiter on duplicates and the limit has been exceeded for the current suspect.

The type of matching (Intraflow or Interflow) determines how express key match results translate to Candidate Match Scores. In Interflow matching, a successful Express Key match always confers a 100 MatchScore onto the Candidate. On the other hand, in Intraflow matching, the score a Candidate gains as a result of an Express Key match depends on whether the record to which that Candidate matched was a match of some other Suspect—Express Key duplicates of a Suspect will always have MatchScores of 100, whereas Express Key duplicates of another Candidate (which was a duplicate of a Suspect) will inherit the MatchScore (not necessarily 100) of that Candidate

Options

1. In the **Load match rule** field, select one of the predefined match rules which you can either use as-is or modify to suit your needs. If you want to create a new match rule without using one of the predefined match rules as a starting point, click **New**. You can only have one custom rule in a dataflow.

Note: The Dataflow Options feature in Enterprise Designer enables the match rule to be exposed for configuration at runtime.

2. Click **Group By** to select a field to use for grouping records in the match queue. Intraflow Match only attempts to match records against other records in the same match queue.
3. Select the **Sort** box to perform a pre-match sort of your input based on the field selected in the **Group By** field.
4. Click **Advanced** to specify additional sort performance options.

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:

$$\frac{(NumberOfRecords \times 2)}{InMemoryRecordLimit} = NumberOfTempFilesN$$

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: $(InMemoryRecordLimit \times MaxNumberOfTempFiles \div 2) \geq TotalNumberOfRecords$

5. Click **Express Match On** to perform an initial comparison of express key values to determine whether two records are considered a match.

Express Key matching can be a useful tool for reducing the number of compares performed and thereby improving execution speed. A loose express key results in many false positive matches. You can generate an express key as part of generating a match key through MatchKeyGenerator. See [Match Key Generator](#) on page 142 for more information.

If two records have an exact match on the express key, the candidate is considered a 100% duplicate. If two records do not match on an express key value, they are compared using the rules-based method.

To determine whether a candidate was matched using an express key, look at the value of the **ExpressKeyIdentified** field, which is either Y for a match or N for no match. Note that suspect records always have an **ExpressKeyIdentified** value of N.

6. In the **Initial Collection Number** text box, specify the starting number to assign to the collection number field for duplicate records.

The collection number identifies each duplicate record in a match queue. Unique records are assigned a collection number of 0. Each duplicate record is assigned a collection number starting with the value specified in the **Initial Collection Number** text box.

7. Select one of the following:

Option	Description
<p>Compare suspect to all candidates</p>	<p>This option matches the suspect to all candidates in the same match group (group by option) even if a duplicate is already found within the match group. For example:</p> <p>Suspect - John Smith Candidate - Bill Jones Candidate - John Smith Candidate - John Smith</p> <p>In the example, the suspect John Smith would be compared to both John smith candidates.</p> <p>Check the Return Unique Candidates box to return records within a match group from the candidate port that have been identified as unique records.</p>
<p>Stop comparing suspect against candidates after finding <i>n</i> duplicates</p>	<p>This option matches the suspect to all candidates in the same match group (group by option) but stops comparing when the user defined number of duplicates have been identified. For example, if you chose to stop comparing candidates after finding one duplicate and you had this data:</p> <p>Suspect - John Smith Candidate - Bill Jones Candidate - John Smith Candidate - John Smith</p> <p>In the example, the suspect record John Smith would stop comparing within the match group when the first John Smith candidate is identified as a duplicate.</p>

8. Click **Generate Data for Analysis** to generate match results. For more information, see [Analyzing Match Results](#).
9. **Assign collection number 0 to unique records**, checked by default, will assign zeroes as collection numbers to unique records. Uncheck this option to generate collection numbers other than zero for unique records. The unique record collection numbers will be in sequence with any other collection numbers. For example, if your matching dataflow finds five records and the first three records are unique, the collection numbers would be assigned as shown in the first group below. If your matching dataflow finds five records and the last two are unique, the collection numbers would be assigned as shown in the second group below.

Option	Description
Collection Number	Record Type
1	Unique
2	Unique
3	Unique
4	Duplicate/Suspect
4	Duplicate/Suspect
Collection Number	Record Type
1	Duplicate/Suspect
1	Duplicate/Suspect
2	Unique
3	Unique
4	Unique

If you leave this box checked, any unique records found in your dataflow will be assigned a collection number of zero by default.

10. Select the **Return match rule name** option to include the selected match rule name in the stage output.
11. Select **Return detailed match information** if you want detailed match information to be displayed as an output for your match rule. For more information about the output fields, see [Output](#) on page 135.

Note: If you enable this field, it will hinder the overall stage performance.

12. If you are creating a new custom matching rule, see [Building a Match Rule](#) for more information.

13. Click **Evaluate** to evaluate how a suspect record scored against candidate records. For more information, see [Interflow Match](#) on page 130.

Output

Table 14: Interflow Match Output Fields

Field Name	Description / Valid Values
CollectionNumber	Identifies a collection of duplicate records. The possible values are 1 or greater.
ExpressMatchIdentified	Indicates whether the match was obtained using the express match key. The possible values are Yes or No.
HasDuplicates	Identifies whether the record is a duplicate of another record. One of the following: Y The record is a suspect record and has duplicates. N The record is a suspect record and has no duplicates. D The record is a candidate record and is a duplicate of the suspect record. U The record is a candidate record but is not a duplicate of the suspect record.
InterflowSourceType	The possible values are input_port_0 or input_port_1
MatchRecordType	Identifies the type of match record in a collection. The possible values are: suspect The original input record that was flagged as possibly having duplicate records. duplicate A record that is a duplicate of the input record. unique A record that has no duplicates.
MatchInfo	It displays the detailed information of each match record in JSON format, for example, a match record's name, type, state as True or False, and score of the algorithm between 0-100. Note: This field is displayed only for a duplicate match record.

Field Name	Description / Valid Values
<i>MatchInfo.RuleName.IsMatch</i>	Displays the match state for the rule. Note: This field is displayed only for a duplicate match record.
<i>MatchInfo.RuleName.Score</i>	Displays the match score for the rule. The possible values are 0-100, with 0 indicating a poor match and 100 indicating a perfect match. Note: This field is displayed only for a duplicate match record.
<i>MatchInfo.RuleName.RuleNodeName.IsMatch</i>	Displays the match state for each node in the rule hierarchy, only if it has participated in the matching process, else, a blank field is displayed. Note: This field is displayed only for a duplicate match record.
<i>MatchInfo.RuleName.RuleNodeName.Score</i>	Displays the match score for each node in the rule hierarchy, only if it has participated in the matching process, else, a blank field is displayed. Note: This field is displayed only for a duplicate match record.
<i>MatchRuleName</i>	Displays the name of the match rule against which matching is performed.
<i>MatchScore</i>	Identifies the overall score between two records. The possible values are 0-100, with 0 indicating a poor match and 100 indicating an exact match.

Note: The Validate Address and Advanced Matching stages both use the MatchScore field. The MatchScore field value in the output of a dataflow is determined by the last stage to modify the value before it is sent to an output stage. If you have a dataflow that contains Validate Address and Advanced Matching stages and you want to see the MatchScore field output for each stage, use a Transformer stage to copy the MatchScore value to another field. For example, Validate Address produces an output field called MatchScore and then a Transformer stage copies the MatchScore field from Validate Address to a field called AddressMatchScore. When the matcher stage runs it populates the MatchScore field with the value from the matcher and passes through the AddressMatchScore value from Validate Address.

Intraflow Match

Intraflow Match locates matches between similar data records within a single input stream. You can create hierarchical rules based on any fields that have been defined or created in other stages of the dataflow.

Options

1. In the **Load match rule** field, select one of the predefined match rules which you can either use as-is or modify to suit your needs. If you want to create a new match rule without using one of the predefined match rules as a starting point, click **New**. You can only have one custom rule in a dataflow.

Note: The Dataflow Options feature in Enterprise Designer enables the match rule to be exposed for configuration at runtime.

2. Click **Group By** to select a field to use for grouping records in the match queue. Intraflow Match only attempts to match records against other records in the same match queue.
3. Select the **Sort** box to perform a pre-match sort of your input based on the field selected in the **Group By** field.
4. Click **Advanced** to specify additional sort performance options.

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:

$$\frac{(NumberOfRecords \times 2)}{InMemoryRecordLimit} = NumberOfTempFilesN$$

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: $(InMemoryRecordLimit \times MaxNumberOfTempFiles \div 2) \geq TotalNumberOfRecords$

5. Click **Express Match On** to perform an initial comparison of express key values to determine whether two records are considered a match.

You can generate an express key as part of generating a match key through MatchKeyGenerator. See [Match Key Generator](#) on page 142 for more information.

6. In the **Initial Collection Number** text box, specify the starting number to assign to the collection number field for duplicate records.

The collection number identifies each duplicate record in a match queue. Unique records are assigned a collection number of 0. Each duplicate record is assigned a collection number starting with the value specified in the **Initial Collection Number** text box.

7. Click **Sliding Window** to enable this matching method. For more information about Sliding Window, see [Sliding Window Matching Method](#) on page 140
8. Click **Generate Data for Analysis** to generate match results. For more information, see [Analyzing Match Results](#).
9. **Assign collection number 0 to unique records**, checked by default, will assign zeroes as collection numbers to unique records. Uncheck this option to generate collection numbers other than zero for unique records. The unique record collection numbers will be in sequence with any other collection numbers. For example, if your matching dataflow finds five records and the first three records are unique, the collection numbers would be assigned as shown in the first group below. If your matching dataflow finds five records and the last two are unique, the collection numbers would be assigned as shown in the second group below.

Option	Description
Collection Number	Record Type
1	Unique
2	Unique
3	Unique
4	Duplicate/Suspect

Option	Description
4	Duplicate/Suspect
Collection Number	Record Type
1	Duplicate/Suspect
1	Duplicate/Suspect
2	Unique
3	Unique
4	Unique

If you leave this box checked, any unique records found in your dataflow will be assigned a collection number of zero by default.

10. Select the **Return match rule name** option to include the selected match rule name in the stage output.
11. Select **Return detailed match information** if you want detailed match information to be displayed as an output for your match rule. For more information about the output fields, see [Output](#) on page 140.

Note: If you enable this field, it will hinder the overall stage performance.

12. For information about modifying the other options, see [Building a Match Rule](#).
13. Click **Evaluate** to evaluate how a suspect record scored against candidate records. For more information, see [Interflow Match](#) on page 130.

Default Matching Method

Using group by (match group) set by the user, the matcher identifies groups of records that might potentially be duplicates of one another. The matcher then proceeds through each record in the group; if the record matches an existing Suspect, the record is considered a Duplicate of that suspect, assigned a Score, CollectionNumber, and MatchRecordType (Duplicate), and eliminated from the match. If, on the other hand, the record matches no existing Suspect within the match group, the record becomes a new Suspect, in that it is added to the current Match group so that it can be matched against by subsequent records. When the matcher has exhausted all records in the current Match group, it eliminates all Suspects from the match, labeling the Match Record type as Unique and assigning a collection number of 0. Those Suspects with a least one duplicate will retain a Match

Record Type of Suspect and is assigned the same collection number as its matched duplicate record. Finally, when all records within a match group have been written to the output. A new match group is compared.

Note: The Default Matching Method will only compare records that are within the same match group.

The type of matching (Intraflow or Interflow) determines how express key match results translate to Candidate Match Scores. In Interflow matching, a successful Express Key match always confers a 100 MatchScore onto the Candidate. On the other hand, in Intraflow matching, the score a Candidate gains as a result of an Express Key match depends on whether the record to which that Candidate matched was a match of some other Suspect—Express Key duplicates of a Suspect will always have MatchScores of 100, whereas Express Key duplicates of another Candidate (which was a duplicate of a Suspect) will inherit the MatchScore (not necessarily 100) of that Candidate

Sliding Window Matching Method

The sliding window algorithm is an algorithm which sequentially fills a pre determined buffer size called a window with the corresponding amount of data rows. As each row is added to the window it's compared to each item already contained in the window. If a match with an item is determined then both the driver record (the new item to add to the window) and the candidates (items already in the window) is given the same group ID. This comparison is continued until the driver record has been compared to all items contained within the window.

As new drivers are added the window will eventually reach its predetermined capacity. At this point the window will slide, hence the term Sliding Window. Sliding simply means that the window buffer will remove and write the oldest item in the window as it adds the newest driver record to the window.

Output

Table 15: Intraflow Match Output

Field Name	Description / Valid Values
CollectionNumber	Identifies a collection of duplicate records. The possible values are 1 or greater.
ExpressMatchIdentified	Indicates whether the match was obtained using the express match key. The possible values are Yes or No.

Field Name	Description / Valid Values
MatchRecordType	<p>Identifies the type of match record in a collection. The possible values are:</p> <p>Suspect A record that other records are compared to in order to determine if they are duplicates of each other. Each collection has one and only one suspect record.</p> <p>Duplicate A record that is a duplicate of the suspect record.</p> <p>Unique A record that has no duplicates.</p>
MatchInfo	<p>It displays the detailed information of each match record in JSON format, for example, a match record's name, type, state as True or False, and score of the algorithm between 0-100.</p> <p>Note: This field is displayed only for a duplicate match record.</p>
MatchInfo.RuleName.IsMatch	<p>Displays the match state for the rule.</p> <p>Note: This field is displayed only for a duplicate match record.</p>
MatchInfo.RuleName.Score	<p>Displays the match score for the rule. The possible values are 0-100, with 0 indicating a poor match and 100 indicating a perfect match.</p> <p>Note: This field is displayed only for a duplicate match record.</p>
MatchInfo.RuleName. RuleNodeName.IsMatch	<p>Displays the match state for each node in the rule hierarchy, only if it has participated in the matching process, else, a blank field is displayed.</p> <p>Note: This field is displayed only for a duplicate match record.</p>
MatchInfo.RuleName. RuleNodeName.Score	<p>Displays the match score for each node in the rule hierarchy, only if it has participated in the matching process, else, a blank field is displayed.</p> <p>Note: This field is displayed only for a duplicate match record.</p>
MatchRuleName	<p>Displays the name of the match rule against which matching is performed.</p>
MatchScore	<p>Identifies the overall score between two records. The possible values are 0-100, with 0 indicating a poor match and 100 indicating an exact match.</p>

Note: The Validate Address and Advanced Matching stages both use the MatchScore field. The MatchScore field value in the output of a dataflow is determined by the last stage to modify the value before it is sent to an output stage. If you have a dataflow that contains Validate Address and Advanced Matching stages and you want to see the MatchScore field output for each stage, use a Transformer stage to copy the MatchScore value to another field. For example, Validate Address produces an output field called MatchScore and then a Transformer stage copies the MatchScore field from Validate Address to a field called AddressMatchScore. When the matcher stage runs it populates the MatchScore field with the value from the matcher and passes through the AddressMatchScore value from Validate Address.

Match Key Generator

Match Key Generator creates a non-unique key for each record, which can then be used by matching stages to identify groups of potentially duplicate records. Match keys facilitate the matching process by allowing you to group records by match key and then only comparing records within these groups.

The match key is created using rules you define and is comprised of input fields. Each input field specified has a selected algorithm that is performed on it. The result of each algorithm is then concatenated to create a single match key field.

In addition to creating match keys, you can also create express match keys to be used later in the dataflow by an Intraflow Match stage or an Interflow Match stage.

You can create multiple match keys and express match keys.

For example, if the incoming record is:

First Name - Fred
 Last Name - Mertz
 Postal Code - 21114-1687
 Gender Code - M

And you define a match key rule that generates a match key by combining data from the record like this:

Input Field	Start Position	Length
Postal Code	1	5
Postal Code	7	4
Last Name	1	5
First Name	1	5

Input Field	Start Position	Length
Gender Code	1	1

Then the key would be:
211141687MertzFredM

Input

The input is any field in the source data.

Options

To define Match Key Generator options click the **Add** button. The **Match Key Field** dialog displays.

Note: The Dataflow Options feature in Enterprise Designer enables Match Key Generator to be exposed for configuration at runtime.

Table 16: Match Key Generator Options

Option Name	Description and Valid Values
-------------	------------------------------

Algorithm	
-----------	--

Option Name	Description and Valid Values
-------------	------------------------------

Specifies one of these algorithms to use to generate the match key:

Consonant	Returns specified fields with consonants removed.
Double Metaphone	Returns a code based on a phonetic representation of their characters. Double Metaphone is an improved version of the Metaphone algorithm, and attempts to account for the many irregularities found in different languages.
Koeln	Indexes names by sound as they are pronounced in German. Allows names with the same pronunciation to be encoded to the same representation so that they can be matched, despite minor differences in spelling. The result is always a sequence of numbers; special characters and white spaces are ignored. This option was developed to respond to limitations of Soundex.
MD5	A message digest algorithm that produces a 128-bit hash value. This algorithm is commonly used to check data integrity.
Metaphone	Returns a Metaphone coded key of selected fields. Metaphone is an algorithm for coding words using their English pronunciation.
SpanishMetaphone	Returns a Metaphone coded key of selected fields for the Spanish language. This metaphone algorithm codes words using their Spanish pronunciation.
Metaphone 3	Improves upon the Metaphone and Double Metaphone algorithms with more exact consonant and internal vowel settings that allow you to produce words or names more or less closely matched to search terms on a phonetic basis. Metaphone 3 increases the accuracy of phonetic encoding to 98%. This option was developed to respond to limitations of Soundex.
Nysiis	Phonetic code algorithm that matches an approximate pronunciation to an exact spelling and indexes words that are pronounced similarly. Part of the New York State Identification and Intelligence System. Say, for example, that you are looking for someone's information in a database of people. You believe that the person's name sounds like "John Smith", but it is in fact spelled "Jon Smyth". If you conducted a search looking for an exact match for "John Smith" no results would be returned. However, if you index the database using the NYSIIS algorithm and search using the NYSIIS algorithm again, the correct match will be returned because both "John Smith" and "Jon Smyth" are indexed as "JAN SNATH" by the algorithm.
Phonix	Preprocesses name strings by applying more than 100 transformation rules to single characters or to sequences of several characters. 19 of those rules are applied only if the characters are at the beginning of the string, while 12 of the rules are applied only if they are at the middle of the string, and 28 of the rules are applied only if they are at the end of the string. The

Option Name	Description and Valid Values
	transformed name string is encoded into a code that is comprised by a starting letter followed by three digits (removing zeros and duplicate numbers). This option was developed to respond to limitations of Soundex; it is more complex and therefore slower than Soundex.
Sonnex	This algorithm determines the similarity between two French-language strings based on the phonetic representation of their characters. It returns a Sonnex coded key of the selected fields.
Soundex	Returns a Soundex code of selected fields. Soundex produces a fixed-length code based on the English pronunciation of a word.
Substring	Returns a specified portion of the selected field.
Field name	Specifies the field to which you want to apply the selected algorithm to generate the match key. For example, if you select a field called LastName and you choose the Soundex algorithm, the Soundex algorithm would be applied to the data in the LastName field to produce a match key.
Start position	Specifies the starting position within the specified field. Not all algorithms allow you to specify a start position.
Length	Specifies the length of characters to include from the starting position. Not all algorithms allow you to specify a length.
Remove noise characters	Removes all non-numeric and non-alpha characters such as hyphens, white space, and other special characters from an input field.
Sort input	Sorts all characters in an input field or all terms in an input field in alphabetical order. Characters Sorts the characters values from an input field prior to creating a unique ID. Terms Sorts each term value from an input field prior to creating a unique ID.

If you add multiple match key generation algorithms, you can use the **Move Up** and **Move Down** buttons to change the order in which the algorithms are applied.

Generating an Express Match Key

Enable the **Generate Express Match Key** option and click **Add** to define an express match key to be used later in the dataflow by an Intraflow Match stage or an Interflow Match stage.

If the **Generate Express Match Key** option is enabled and the **Express match key on** option is selected in a downstream Interflow Match stage or Intraflow Match stage, the match attempt is first made using the express match key created here. If two records' express match keys match, then the record is considered a match and no further processing is attempted. If the records' express match keys do not match, then the match rules defined in Interflow Match or Intraflow Match are used to determine if the records match.

Output

Table 17: Match Key Generator Output

Field Name	Description / Valid Values
ExpressMatchKey	A value indicating the match level. If the express match key is a match, the score is 100. If the express match key does not match, then a score of 0 is returned.
MatchKey	The key generated to identify records.

Private Match

Private Match enables two entities to compare datasets and identify common records without compromising sensitive information. For example, two companies could be interested in launching a joint marketing campaign. Each company has its own database containing customer information, and the companies want to determine which customers shop at both companies to use a more targeted approach in the campaign. However, to ensure data security and comply with privacy regulations, the companies do not wish to share these databases with each other or to give them to a third party to conduct a match. The private match feature makes it possible for the two databases to be matched against each other without breaching security or breaking privacy laws.

Private Match is used in one of three modes:

- **Encrypt mode**—The first user inputs his data, and an index field and match field are extracted and encrypted. A public key and a displacement table containing the first user's data are generated for the second user, and a private key is generated for the first user to use later.
- **Private Match mode**—The second user inputs his data and the first user's encrypted data, provides the public key and displacement table, and performs a match. A file containing the matched data is generated to be sent to the first user.

- Decrypt mode—The first user inputs the second user's encrypted data, provides the private key, and generates output containing a matched index of both user's data.

By using the encrypt function (Encrypt mode) the security is retained while a match function is performed (Private Match mode), and then a decrypt function shows the output of the matched data (Decrypt mode). All files generated and shared between users are encrypted and unreadable.

Input

Input requirements for the Private Match stage vary depending on the task you are performing:

- Encrypt mode—A file containing the first user's data must be attached to the input port. This can be a text file, database, or almost any kind of source file.
- Private Match mode—A file containing the second user's data must be attached to the first input port; this can also be a text file, database, or almost any kind of source file. The encrypted data generated by the first user must be attached to the second input port.
- Decrypt mode—The output file generated by the second user.

Options

Options for the Private Match stage vary depending on the task you are performing.

Note: If you upgrade to version 11.0 or later from version 10.x and you produced private keys, you will need to regenerate those keys because keys generated in version 10.x are not compatible with versions 11.0 and later.

Encrypt Mode

1. Select the **Encrypt** operation.
2. Select the **index field** that provides a unique ID for each record in the file. The unique ID must be a numeric value.
3. Select the **match field** that should be used to match against the second user's data.
4. Specify the path to and name of the **Public key** file that will be created when you run the job.
5. Specify the path to and name of the **Key Store** file that will be created when you run the job.
6. Specify the path to and name of the **Displacement table** file that will be created when you run the job.
7. Enter a name for the **output column** that will contain the encrypted data in the output file that is sent to the second user.
8. Press **OK**.

Private Match Mode

1. Select the **Private Match** operation.
2. Select the **index field** that provides a unique ID for each record in the file. The unique ID must be a numeric value.

3. Select the **match field** that should be used to match against the first user's data.
4. Enter the name of the **encrypted data field** that the first user entered in step 7 of the Encrypt Mode instructions.
5. Navigate to the **Public key** file.
6. Navigate to the **Displacement table** file.
7. Enter a name for the **output column** that will contain the encrypted data in the output file that is sent to the first user.

Decrypt Mode

1. Select the **Decrypt** operation.
2. Select the **encrypted data field** entered in step 7 of the Private Match Mode instructions.
3. Navigate to the **Key Store** file.
4. Select the **output column** that will contain the decrypted data in the output file. The format of the data in this field is the matched index of the first user's data and the matched index of the second users' data, separated by a comma character (,), as in the following:

```
User1Data,User2Data
```

Output

Output requirements for the Private Match stage vary depending on the task you are performing:

- Encrypt mode—A file generated by the Write to File stage that contains the first user's encrypted data.
- Private Match mode—A file generated by the Write to File stage that contains encrypted information about the match results.
- Decrypt mode—A file generated by the Write to File stage that contains the matched index of both users' data.

Transactional Match

Transactional Match matches suspect records against candidate records that are returned from the Candidate Finder stage. Transactional Match uses matching rules to compare the suspect record to all candidate records with the same candidate group number (assigned in Candidate Finder) to identify duplicates. If the candidate record is a duplicate, it is assigned a collection number, the match record type is labeled a Duplicate, and the record is then written out. Any unmatched candidates in the group are assigned a collection number of 0, labeled as Unique and then written out as well.

Note: Transactional Match only matches suspect records to candidates. It does not attempt to match suspect records to other suspect records as is done in Intraflow Match.

Options

1. In the **Load match rule** field, select one of the predefined match rules which you can either use as-is or modify to suit your needs. If you want to create a new match rule without using one of the predefined match rules as a starting point, click **New**. You can only have one custom rule in a dataflow.

Note: The Dataflow Options feature in Enterprise Designer enables the match rule to be exposed for configuration at runtime.

2. Select **Return unique candidates** if you want unique candidate records to be included in the output from the stage.
3. Select **Generate data for analysis** if you want to use the Match Analysis tool to analyze the results of the dataflow. For more information, see [Analyzing Match Results](#).
4. Select **Return detailed match information** if you want detailed match information to be displayed as an output for your match rule.

Note: For detailed information about the output fields, see [Output](#) on page 150.

5. For information about modifying the other options, see [Building a Match Rule](#).
6. Click **Evaluate** to evaluate how a suspect record scored against candidate records. For more information, see [Interflow Match](#) on page 130.

Output

Table 18: Transactional Match Output

Field Name	Description / Valid Values
HasDuplicates	<p>Identifies whether the record is a duplicate of another record. One of the following:</p> <p>Y The record is a suspect record and has duplicates.</p> <p>N The record is a suspect record and has no duplicates.</p> <p>D The record is a candidate record and is a duplicate of the suspect record.</p> <p>U The record is a candidate record but is not a duplicate of the suspect record.</p>
DuplicateCount	It displays the number of duplicates within a match group.

Field Name	Description / Valid Values
MatchGroupSize	It determines the size and displays the number of records in a match group.
MatchRecordType	<p>Identifies the type of match record in a collection. The possible values are:</p> <p>Suspect The original input record that was flagged as possibly having duplicate records.</p> <p>Duplicate A record that is a duplicate of the input record.</p> <p>Unique A record that has no duplicates.</p>
MatchInfo	It displays the detailed information of each match record in JSON format, for example, a match record's name, type, state as True or False, and score between 0-100.
MatchScore	Identifies the overall score between two records. The possible values are 0-100, with 0 indicating a poor match and 100 indicating an exact match.
MatchInfo.RuleName.IsMatch	<p>Displays the match state for the rule as <code>True</code> or <code>False</code>, with <code>True</code> representing a match and <code>False</code> indicating no matches. This information is displayed if you select the Return detailed match information option in the Transactional Match Options screen.</p> <p>Note: The value in this field is derived from the match state of the child nodes based on the defined parent rule configuration.</p>
MatchInfo.RuleName.Score	<p>Displays the match score for the rule. The possible values are 0-100, with 0 indicating a poor match and 100 indicating a perfect match. This information is displayed if you select the Return detailed match information option in the Transactional Match Options screen.</p> <p>Note: The score in this field is derived from the scores of the child nodes on the basis of the defined parent rule configuration.</p>

Field Name	Description / Valid Values
<code>MatchInfo.RuleName.RuleNodeName.IsMatch</code>	<p>Displays the match state for each node in the rule hierarchy as <code>True</code> or <code>False</code>, with <code>True</code> representing a match and <code>False</code> indicating no matches. <code>RuleNodeName</code> is a variable, which is replaced by the hierarchical node names in your match rules.</p> <p>Each node in the rule hierarchy outputs this field, if you have selected the Return detailed match information option in the Transactional Match Options screen.</p>
<code>MatchInfo.RuleName.RuleNodeName.Score</code>	<p>Displays the match score for each node in the rule hierarchy. <code>RuleNodeName</code> is a variable, which is replaced by the hierarchical node names in your match rules.</p> <p>Each node in the rule hierarchy outputs this field, if you have selected the Return detailed match information option in the Transactional Match Options screen.</p> <p>The possible values are 0-100, with 0 indicating a poor match and 100 indicating a perfect match.</p>

Note: The Validate Address and Advanced Matching stages both use the MatchScore field. The MatchScore field value in the output of a dataflow is determined by the last stage to modify the value before it is sent to an output stage. If you have a dataflow that contains Validate Address and Advanced Matching stages and you want to see the MatchScore field output for each stage, use a Transformer stage to copy the MatchScore value to another field. For example, Validate Address produces an output field called MatchScore and then a Transformer stage copies the MatchScore field from Validate Address to a field called AddressMatchScore. When the matcher stage runs it populates the MatchScore field with the value from the matcher and passes through the AddressMatchScore value from Validate Address.

Write to Search Index

Write to Search Index enables you to create a full-text index based on the data coming in to the stage. Having this data in a dedicated search index results in quicker response time when you conduct searches against the index from other Spectrum Technology Platform stages. Full-text-search indexes are preferable to relational databases when you have a great deal of free-form text data that needs to be searched or categorized or if you support a high volume of interactive, text-based queries.

Write to Search Index uses an analyzer to break input text into small indexing elements called tokens. It then extracts search index terms from those tokens. The type of analyzer used—the manner in which input text is broken into tokens—determines how you will then be able to search for that text. For example, the *Keyword* analyzer always does an exact match and tokenizes the whole string as single token, while the *Standard* analyzer breaks the string to create tokens. The *Stop Word* analyzer

is all the more sophisticated and removes articles, such as "a" or "the" from the string before tokenizing, thus shrinking the index size.

Search indexes support the near real time feature, allowing indexes to be updated almost immediately, without the need to close and rebuild the stage using the search index.

For information about the search index tasks you can perform through the command line, see Search Indexes in [Administration Utility](#) section of the Administration Guide.

Options

- In Enterprise Designer, double-click the Write to Search Index stage on the canvas.
- Enter a **Name** for the index.
- Select a **Write mode**. When you regenerate an index, you have options related to how the new data should affect the existing data.
 - Create or Overwrite**—New data will overwrite the existing data and the existing data will no longer be in the index.
 - Update or Append**—New data will overwrite existing data, and any new data that did not previously exist will be added to the index.
 - Append**—New data will be added to the existing data and the existing data will remain in tact.
 - Delete**—Data for the selected field will be deleted from the search index.
- Select the **Key field** on the basis of which you want to **Update or Append** or **Delete** the records.
 - In case of **Create or Overwrite** mode, the **Key field** needs to be unique for Elastic search indexes (used in a distributed environment). If you leave the field blank, all the records get stored in the index irrespective of any duplication. However, you will not be able to perform any write operation, such as update, append, and delete on this index. The following table explains the indexing behavior if the **Key field** is non-unique for Lucene and Elastic search indexes.

Write mode	Key field	Lucene search index	Elastic search index
Create or Overwrite	Duplicate records with same Key field	All the records are stored. Note: The duplicate records with same key field get overwritten as soon as you run the update operation.	All duplicate records with the same Key field are overwritten .
Update or Append	Duplicate records with same Key field	Duplicates are overwritten.	Duplicates are overwritten.

5. Check the **Batch commit** box if you want to specify the number of records to commit in a batch while creating the search index. Then enter that number in the **Batch size** field. Default is 5000.
6. Select an **Analyzer** to build:
 - **Standard**—Provides a grammar-based tokenizer that contains a superset of the Whitespace and Stop Word analyzers. Understands English punctuation for breaking down words, knows words to ignore (via the Stop Word Analyzer), and performs technically case-insensitive searching by conducting lowercase comparisons. For example, the string “Precisely Software” would be returned as two tokens: “precisely” and “software”. For a comparison of Standard and Keyword analyzers, see [Standard and Keyword Analyzer](#) on page 156.
 - **Whitespace**—Separates tokens with whitespace. Somewhat of a subset of the Standard Analyzer in that it understands word breaks in English text based on spaces and line breaks.
 - **Stop Word**—Removes articles such as “the,” “and,” and “a” to shrink the index size and increase performance.
 - **Keyword**—Creates a single token from a stream of data and keeps it as is. For example, the string “Precisely Software” would be returned as just one token “Precisely Software”. For a comparison of Standard and Keyword analyzers, see [Standard and Keyword Analyzer](#) on page 156.
 - **Russian**—Supports Russian-language indexes and type-ahead services. Also supports many stop words and removes articles such as “and,” “I,” and “you” to shrink the index size and increase performance.
 - **German**—Supports German-language indexes and type-ahead services. Also supports many stop words and removes articles such as “the,” “and,” and “a” to shrink the index size and increase performance.
 - **Danish**—Supports Danish-language indexes and type-ahead services. Also supports many stop words and removes articles such as “at,” “and,” and “a” to shrink the index size and increase performance.
 - **Dutch**—Supports Dutch-language indexes and type-ahead services. Also supports many stop words and removes articles such as “the,” “and,” and “a” to shrink the index size and increase performance.
 - **Finnish**—Supports Finnish-language indexes and type-ahead services. Also supports many stop words and removes articles such as “is,” “and,” and “of” to shrink the index size and increase performance.
 - **French**—Supports French-language indexes and type-ahead services. Also supports many stop words and removes articles such as “the,” “and,” and “a” to shrink the index size and increase performance.
 - **Hungarian**—Supports Hungarian-language indexes and type-ahead services. Also supports many stop words and removes articles such as “the,” “and,” and “a” to shrink the index size and increase performance.
 - **Italian**—Supports Italian-language indexes and type-ahead services. Also supports many stop words and removes articles such as “the,” “and,” and “a” to shrink the index size and increase performance.

- Norwegian—Supports Norwegian-language indexes and type-ahead services. Also supports many stop words and removes articles such as "the" "and," and "a" to shrink the index size and increase performance.
 - Portuguese—Supports Portuguese-language indexes and type-ahead services. Also supports many stop words and removes articles such as "the" "and," and "a" to shrink the index size and increase performance.
 - Spanish—Supports Spanish-language indexes and type-ahead services. Also supports many stop words and removes articles such as "the" "and," and "a" to shrink the index size and increase performance.
 - Swedish—Supports Swedish-language indexes and type-ahead services. Also supports many stop words and removes articles such as "the" "and," and "a" to shrink the index size and increase performance.
 - Hindi—Supports Hindi-language indexes and type-ahead services. Also supports many stop words and removes articles such as "by" "and," and "a" to shrink the index size and increase performance.
7. To update the analyzer of all fields present in the list, Select an analyzer from **update Analyzers to...** drop down.
 8. To reload the schema from the server, click **Reload Schema**.

Note: You can change the field name by typing the new name directly in the **Fields** column. However, you cannot change the **Stage Fields** name or the **Type**.
 9. To selectively add/remove fields from your input source, click **Quick Add**. The **Quick Add** pop-up window displays a list of all the fields from the input source. Select the fields that you want to add and click **OK**.
 10. Select the field(s) whose data you want to store. For example, using an input file of addresses, you could index just the Postal Code field but choose to store the remaining fields (such as Address Line 1, City, State) so the entire address is returned when a match is found using the index search.
 11. Select the field(s) whose data you want to be added to the index for a search query.

Note: If you want to delete certain fields, select those and click **Delete**.
 12. If necessary, change the analyzer for any field that should use something other than what you selected in the Analyzer field.
 13. Click **OK**.

Output

Write to Search Index has one output port, called **Error Port**, which collects data for records that could not be processed and added to the search index. Records that pass through the error port into the sink are considered malformed.

Records are forwarded to the error port if the value of the **KeyField** in the input record is blank or null. For the search index engines that support standalone searches (the legacy index engine), the error port functions only for the Update and Delete operations. However, for distributed support index engine, it collects malformed records for all the four operations: Create, Update, Delete, and Append.

Search Index Management

The **Search Index Management** tool enables you to perform the following tasks:

- Add/remove fields in an existing search index
- Delete one or more search indexes

To add/remove fields to an existing search index:

1. Select **Tools > Search Index Management**. The **Search Index Management** pop-up window is displayed showing the existing search indexes.
2. Select the index to be modified, and click **Modify**. The **Modify Index** page is displayed showing a list of all the fields in the search index with their details.
3. To delete a field, select it from the list, and click **Remove field**.
4. To add a new field to the list, click **Add field**. The **Add Field** pop-up window is displayed, which allows you to add the following details:
 - **Field name**: Enter the name of the field.
 - **Type**: Select the field type. The options are: string, integer, long, float, and double.
 - **Index**: Select the check-box to add it to the index for search query.
 - **Store**: Select the check-box to store the field.
 - **Analyzer**: From the list of options, select the analyzer you want to use for indexing the field.

Note: This option will be enabled only if you have selected to index this field.

To delete an existing search index:

1. Select **Tools > Search Index Management**.
2. Select the search indexes you want to delete.
3. Click **Delete**.
4. Click **Close**.

Note: You can also delete a search index by using the Administration Utility. The command is `index delete --d IndexName`, where *IndexName* is the name of the index you want to delete.

Standard and Keyword Analyzer

Before choosing between the Standard and Keyword analyzers you should be aware of the following difference in the behavior of these two analyzers:

- **Standard analyzer**: It breaks the string to tokenize it and also converts all its tokens to lower case.

- **Keyword analyzer:** It tokenizes the whole string and keeps it as is (does not change the case).

Example:

Let us take an example of *contains any* algorithm and assume the input is P O. In this case, the **Standard** analyzer will return both P O and P records, while the **Keyword** analyzer will return only the P O records (shown below).

Search result with Standard analyzer:

Point		Point 2					
SearchTerm	CandidateGroup	HasDuplicates	Index	IndexField	Term	TransactionRecordType	
P O	1	Y				Suspect	
P O	1	D	P O	4	Test4	Candidate	
P O	1	D	P	1	Test 1	Candidate	

Search result with Keyword analyzer:

Point		Point 2					
SearchTerm	CandidateGroup	HasDuplicates	Index	IndexField	Term	TransactionRecordType	
P O	1	Y				Suspect	
P O	1	D	P O	4	Test4	Candidate	

Analytics Scoring stages

Binning Lookup

Introduction to Binning Lookup

Binning Lookup applies previously defined binning to new data using existing bins created in dataflows using the Machine Learning **Binning** stage.

Defining Binning Properties

1. Under **Primary Stages > Deployed Stages > Analytics Scoring**, click the **Binning Lookup** stage and drag it onto the canvas, placing it where you want on the dataflow and connecting it to other stages.

Note: The input stage must contain the data that you want binned. One output stage is required for the binned output; you may optionally connect a second output stage to capture the binning summary.

2. Select the appropriate **Binning name** from the drop-down.
These are names of existing bins that were created by a dataflow that uses the Machine Learning Binning stage.
The **Binning type** and **Description** fields are imported with the bins from the selected Binning name and are therefore noneditable.

3. The **Inputs** grid shows each field that was included for binning in the Binning stage along with the data type.
4. Click **OK** to save your settings.

Binning Output

This tab shows the fields and data types that are being binned by the Binning Lookup stage. You can edit entries in the **Spectrum Binned Fields** column to enter a new name for a binned field. In the **Include** column, you can select or clear check boxes to include or exclude binned fields. For more information about output generated by a binning stage, see [Binning Output \(Binning Stage\)](#) in the *Machine Learning Guide*.

Java Model Scoring

Introduction to Model Scoring

Model Scoring enables you to score new data using the formula created when you fit a machine learning model.

Note: Models must first be exposed through the web interface before they will appear in the Model Scoring stage.

To score your data, you must complete two tabs of the Model Scoring Options dialog. First identify the model and its type, and then ensure the model's fields are correctly mapped to Spectrum Technology Platform fields. Following that, you configure the output by selecting which fields you want to include and running your job. The output will appear on the **Model Output** tab.

After running your job, you will need to send the scores to an output table and then run the dataflow or web service.

Defining Model Properties

1. Under **Primary Stages / Deployed Stages / Analytics Scoring**, click the **Model Scoring** stage and drag it onto the canvas, placing it where you want on the dataflow and connecting it to input and output stages.
2. Double-click the Model Scoring stage to show the **Model Scoring Options** dialog box.
3. Optional: Select the type of a model you are scoring in the **Type filter** drop-down.
4. Select the **Type filter** being used to score the model.
5. Select the **Model name** from the drop-down.
6. Enter the type of model you are scoring in the **Model type** field.
7. Optional: Enter a **Description** of the model.
8. The **Inputs** table shows information for the model's input fields. These fields and their data types automatically map to Spectrum fields and data types, but if necessary you can manually change the mappings.
9. Click **OK** to save these options or continue to the next tab.

Configuring Model Output

The **Outputs** table shows information for the model's output fields. These fields and their data types automatically map to Spectrum fields and data types, but if necessary you can manually change the mappings.

1. Click **Include** for each field whose data you want included in the model's output.
2. Click **OK** to save the model.

PMML Model Scoring

Introduction to PMML Model Scoring

The PMML Model Scoring stage is capable of evaluating analytical models which have been published to the Analytics Scoring Repository in the context of a dataflow. The evaluator operates on single data rows using the fields from each row as the inputs to the model. User selected outputs from the model are written to the output channel.

Note: For details of the supported model types and type mappings see [Supported Model Formats](#) on page 162

Deploying a Model

This procedure describes how to configure the PMML Model Scoring stage to deploy an analytics model as part of a dataflow.

1. Under **Primary Stages > Deployed Stages > Analytics Scoring**, click the **PMML Model Scoring** stage and drag it onto the canvas, placing it where you want on the dataflow and connecting it to input and output stages.
2. Double click the PMML Model Scoring stage to show the **PMML Model Scoring Options** dialog box.

By default the options dialog shows the details of the first model in the list of available models.

3. Click the **Type Filter** drop-down and select the model type to filter by.
Only model types which are associated with at least one model in the **Analytics Scoring Repository** are listed.
4. Click the **Model** drop-down and select the model to deploy.
The details of the chosen model are displayed in the dialog.

Model type	The type of the selected model as described in Supported Model Formats on page 162.
Description	A short text about the purpose of the model.
Inputs	A table containing information about all the required input fields for the model. Each row contains information about an input field. The model input field name will automatically be mapped to a valid Spectrum field name on publish (see

Add a Model in a Web Browser Client or **Add a Model via Spectrum Miner Spectrum Connector**).

Model Field Name	The name of the field as specified in the model definition.
Spectrum Field Name	The name of the field as used in the Spectrum platform.
Model Field Type	the type of the field as specified in the model definition.
Spectrum Field Type	The Spectrum field type that is mapped to the model field type as described in QMML on page 163 and PMML on page 164.

5. Select the **Configuration** tab.

Details about the model's **Outputs** are displayed in a table. It contains information about all the output fields for the model. Each row has information about an output field. The model output field name will automatically be mapped to a valid Spectrum field name on publish (see **Add a Model in a Web Browser Client** or **Add a Model via Spectrum Miner Spectrum Connector**).

Model Field Name	The name of the field as specified in the model definition.
Spectrum Field Name	The name of the field as used in the Spectrum platform.
Model Field Type	The type of the field as specified in the model definition.
Spectrum Field Type	The Spectrum field type that is mapped to the model field type as described in QMML on page 163 and PMML on page 164.
Include	A check box to specify whether to use this output.

6. Optional: Uncheck the **Included** column of any row in the **Outputs** table to exclude the output (stop it from being written to the output channel).

At least one output must remain selected. If all outputs are excluded then a validation error symbol will appear beside the **Outputs** table. This means the current model configuration is invalid and the model cannot be deployed. The validation error symbol will remain visible until the error is corrected.

7. Optional: Click the **Spectrum Field Name** column of any row in the **Outputs** table to rename the fields as required.

No two outputs can share the same **Spectrum Field Name** and **Spectrum Field Names** must follow the standard Spectrum Technology Platform field naming conventions. If any validation errors are detected then a validation error symbol is displayed beside the **Outputs** table; hovering the mouse cursor over a validation error symbol shows the error details.

Note: Renaming an output's **Spectrum Field Name** only affects the specific instance of the stage, it does not update the Analytics Scoring Repository .

Note: Spectrum Technology Platform field names must:

- Be unique.
- Be non-empty.
- Contain only alphanumeric, period or underscore characters.
- Cannot start with a period.

8. Finally click **OK** to save the chosen model and configuration.

Reconfigure PMML Model Scoring Settings

This procedure describes how to re-configure the PMML Model Scoring stage to rename or change the outputs that are generated from the PMML Model Scoring stage or to change the deployed model to be evaluated by the stage.

1. Double click the PMML Model Scoring stage to show the **PMML Model Scoring Options** dialog box.

The options dialog shows the model which was previously selected, configured and deployed.

Note: If the selected model has been deleted from the Analytics Scoring Repository prior to opening the **PMML Model Scoring Options** dialog, a validation error symbol is displayed beside the **Model** drop-down list. This means any new changes made to the model's configuration cannot be deployed. Clicking **Cancel** will exit the **PMML Model Scoring Options** and allow the deleted model to be used within the dataflow as previously configured. In order to apply any new changes to the stage configuration, select a different (non-deleted) model from the **Model** drop-down list. After applying a change of model, further re-configuration of the stage will no longer have the deleted model available in the **Model** drop-down list.

2. Optional: From the **Model** tab select a different model from the **Model** drop-down list to change which model is used within the dataflow.

Note: On changing the selected model in the **Model** drop-down list, all configuration changes for the previously chosen model will be discarded. Re-selecting the model will result in the default output configuration for the model. Clicking **Cancel** will undo any changes made since opening the **PMML Model Scoring Options** allowing any pending changes to be reverted.

Note: On selecting a model, if the model has been deleted since the **PMML Model Scoring Options** dialog was opened then a validation error symbol will be displayed beside the **Model** drop-down list. In this scenario the **Inputs** and **Outputs** will not be available and re-configuration of the model settings will not be able to be applied until a non-deleted model has been selected.

3. Select the **Configuration** tab

4. Make any desired changes to the selected model's outputs configuration.

For example, rename a **Spectrum Field Name**, or change its **Include** check box status to be included/excluded from the dataflow.

Any changes to the outputs must adhere to the following validation rules: At least one output must be included. Output's **Spectrum Field Name** must be unique and must follow the standard Spectrum Technology Platform field naming conventions. If any validation errors are detected then a validation error symbol will be displayed beside the table; hovering the mouse cursor over a validation error symbol will show the error details.

Note: Spectrum Technology Platform field names must:

- Be unique.
- Be non-empty.
- Contain only alphanumeric, period or underscore characters.
- Cannot start with a period.

5. Once all desired changes have been made click the **OK** button to apply the changes.

Output

The PMML Model Scoring stage returns the selected model output fields. Additionally, if the PMML Model Scoring stage fails to process a record, it returns the fields Status, Status.Code, and Status.Description. These fields provide information on why the stage failed to process the record as detailed below.

Field Name	Description				
Status	<p>Reports the success or failure of the evaluation attempt:</p> <table> <tr> <td>null</td> <td>Success</td> </tr> <tr> <td>F</td> <td>Failure</td> </tr> </table>	null	Success	F	Failure
null	Success				
F	Failure				
Status.Code	<p>Reason for failure or error:</p> <table> <tr> <td>InputConversionFailed</td> <td>Failed to convert Spectrum input field types to the required model field types.</td> </tr> <tr> <td>ModelEvaluationFailed</td> <td>Model evaluation failed for the given record.</td> </tr> </table>	InputConversionFailed	Failed to convert Spectrum input field types to the required model field types.	ModelEvaluationFailed	Model evaluation failed for the given record.
InputConversionFailed	Failed to convert Spectrum input field types to the required model field types.				
ModelEvaluationFailed	Model evaluation failed for the given record.				
Status.Description	Description of the problem.				

Supported Model Formats

The PMML Model Scoring supports the deployment of analytical models saved in both **QMML** on page 163 and **PMML** on page 164 file formats.

QMML on page 163 models can be created and exported from Spectrum Miner. All types of analytical models and segmentations exported from Spectrum Miner are supported by the PMML Model Scoring stage and can be deployed within a Spectrum Technology Platform dataflow.

PMML on page 164 models can be created and exported from many commercial and open source modeling tools.

Note: Models may only be published in UTF-8 encoding.

QMML

QMML is a proprietary XML based file format used to represent model results generated from Spectrum Miner.

All types of analytical models and segmentations exported from Spectrum Miner are supported by the PMML Model Scoring stage and can be deployed within a Spectrum Technology Platform dataflow.

Type Mapping

QMML model inputs and outputs are automatically mapped to Spectrum Technology Platform field types.

QMML Field Type	Spectrum Technology Platform Field Type
integer	integer
real	double
string	string
date	datetime

Supported Models

All model types constructed within Spectrum Miner (including decision tree, scorecard, cluster analysis and naive Bayes models) are interpreted by Analytics Scoring as a Miner Model.

Miner Model

A Miner Model is any type of **QMML** on page 163 model exported from Spectrum Miner, such as those generated from the decision tree, scorecard, cluster analysis or naive Bayes modeling tools.

Unsupported Features

All compiled **QMML** on page 163 exported from Spectrum Miner is supported.

Model Outputs

Field	Description
<i>dynamic fields</i>	A field is output for each QMML on page 163 model output.

PMML

Predictive Model Markup Language (PMML) is an XML-based file format developed by the Data Mining Group to provide a way for applications to describe and exchange models produced by data mining and machine learning algorithms. PMML files can be created and exported from many commercial and open source modeling tools.

Type Mapping

PMML model inputs and outputs are automatically mapped to Spectrum Technology Platform field types. All PMML models inputs and outputs must be of supported types.

PMML Field Type	Spectrum Technology Platform Field Type
string	string
integer	integer
float	float
double	double
boolean	boolean
date	date
dateDaysSince[1960]	date
dateDaysSince[1970]	date

PMML Field Type	Spectrum Technology Platform Field Type
dateDaysSince[1980]	date
dateDaysSince[0]	not supported
time	time
timeSeconds	time
dateTime	datetime
dateTimeSecondsSince[1960]	datetime
dateTimeSecondsSince[1970]	datetime
dateTimeSecondsSince[1980]	datetime
dateTimeSecondsSince[0]	not supported

Supported Models

Analytics Scoring currently supports the PMML model types detailed in the following sections.

Association Rule

A **PMML** on page 164 association rule model represents rules where some set of items is associated to another set of items. For example a rule can express that a certain product or set of products is often bought in combination with a certain set of other products, also known as Market Basket Analysis. An Association Rule model typically has two variables: one for grouping records together into transactions and another that uniquely identifies each record.

Model Element

```
<AssociationModel functionName="associationRules" ...
```

Unsupported Features

Non-string field types for the field(s) identifying the item are not supported.

Having more than one field for grouping records is not supported.

Association Rule models with the <MiningSchema> element containing a reference to a <DerivedField> element are not supported.

Model Outputs

Supported Model Output Features	Description
transformedValue	A value generated via a transformation expression applied to the predicted model output.
decision	A value generated via an expression applied to the predicted model output resulting in a categorized value.
entityId	The id of the winning rule (default), or the rule specified by the rank value. If the selected rule does not provide an id, a 1-based index is returned.
ruleId	This is identical to the entityId option and has been deprecated as of PMML 4.2. Although its use is currently supported it is recommended to use entityId.
affinity	The affinity of the winning rule (default), or the rule specified by the rank value.
antecedent	The antecedent of the winning rule (default), or the rule specified by the rank value. This output will be formatted as a comma separated string of values.
consequent	The consequent of the winning rule (default), or the rule specified by the rank value. This output will be formatted as a comma separated string of values.
rule	The winning rule (default), or the rule specified by the rank value. This output will return a description of the rule, formatted in the following way: { <i>antecedent</i> }->{ <i>consequent</i> }.
confidence	The confidence of the winning rule (default), or the rule specified by the rank value.
support	The support of the winning rule (default), or the rule specified by the rank value.
lift	The lift of the winning rule (default), or the rule specified by the rank value.
leverage	The leverage of the winning rule (default), or the rule specified by the rank value.

Clustering

A **PMML** on page 164 clustering model determines the best matching cluster for a given record based on the distance or similarity measure used for clustering. A cluster is a subset of similar data. Clustering (also called unsupervised learning) is the process of dividing a dataset into groups such that the members of each group are as similar to each other as possible and different groups are as dissimilar from each other as possible.

Model Element

```
<ClusteringModel functionName="clustering" ...
```

Unsupported Features

Clustering models with the <MiningSchema> element containing a reference to a <DerivedField> element are not supported.

Model Outputs

Supported Model Output Features	Description
predictedValue	The best matching cluster based on the distance or similarity measure used for clustering.
transformedValue	A value generated via a transformation expression applied to the predicted model output.
decision	A value generated via an expression applied to the predicted model output resulting in a categorized value.
predictedDisplayValue	The human readable value used to represent the predicted value from the model.
entityId	If present, the 1-based index (implicit identifier) of the winning/predicted cluster.
affinity	The value of the distance or the similarity of the provided record to the predicted cluster as defined in the model.

Classification Tree

A **PMML** on page 164 classification tree model predicts membership of a categorical dependent variable from one or more independent variables.

Model Element

```
<TreeModel functionName="classification" ...
```

Unsupported Features

Classification trees with a missing value strategy of "aggregateNodes" or "weightedConfidence" are not supported.

Classification Tree models with the <MiningSchema> element containing a reference to a <DerivedField> element are not supported.

Model Outputs

By default the target field will be available as an output field - this is a synonym for the predictedValue feature.

Supported Model Output Features	Description
predictedValue	The categorical dependent variable that we are predicting membership of.
transformedValue	A value generated via a transformation expression applied to the predicted model output.
decision	A value generated via an expression applied to the predicted model output resulting in a categorized value.
predictedDisplayValue	The human readable value used to represent the predicted value from the model.
probability	The statistical probability of the predicted value. Multiple probability outputs can be specified in the model, one for each predicted category or by rank.
residual	The residual of the probability output value (1 - probability) for the predicted category or rank. Multiple residual outputs can be specified in the model, one for each predicted category or by rank.

Supported Model Output Features	Description
------------------------------------	-------------

entityId	If present, the ID of the tree node of the predicted result.
----------	--

Regression Tree

A PMML regression tree model predicts the value of a numeric-dependent variable (such as the price of a house) from one or more independent variables. It does this by building a decision tree model that is based on one or more predictors.

Model Element

```
<TreeModel functionName="regression" ...
```

Unsupported Features

Regression trees with integer or float target fields are not supported unless a <Targets> element is specified with the appropriate castInteger attribute.

Regression tree models with the <MiningSchema> element containing a reference to a <DerivedField> element are not supported.

Model Outputs

By default the target field will be available as an output field. This is a synonym for the predictedValue feature. Float target fields will always be cast to integer.

Supported Model Output Features	Description
------------------------------------	-------------

predictedValue	The numeric dependent variable that we are predicting.
----------------	--

transformedValue	A value generated via a transformation expression applied to the predicted model output.
------------------	--

decision	A value generated via an expression applied to the predicted model output resulting in a categorized value.
----------	---

entityId	If present, the ID of the tree node of the predicted result.
----------	--

Naive Bayes

A **PMML** on page 164 Naive Bayes model predicts the value of a target from evidence given by one or more predictor fields using Bayes' Theorem. Naive Bayes models require the target field to be discretized so that a finite number of values are considered by the model. Predictor fields may be either discrete or continuous.

Model Element

```
<NaiveBayesModel functionName="classification" ...
```

Unsupported Features

Naive Bayes models with the <MiningSchema> element containing a reference to a <DerivedField> element are not supported.

Model Outputs

By default the target field will be available as an output field - this is a synonym for the predictedValue feature.

Supported Model Output Features	Description
predictedValue	The categorical variable that we are predicting membership of.
transformedValue	A value generated via a transformation expression applied to the predicted model output.
decision	A value generated via an expression applied to the predicted model output resulting in a categorized value.
predictedDisplayValue	The human readable value used to represent the predicted value from the model.
probability	The statistical probability of the predicted value.
residual	The residual of the probability output value (1 - probability) for the predicted value.

Regression

A **PMML** on page 164 regression model predicts the value of a numeric dependent variable from one or more independent variables.

Model Element

```
<RegressionModel functionName="regression" ...
```

Unsupported Features

Regression models with a "normalizationMethod" attribute set to the value "simplemax", "probit", "cloglog" or "loglog" are not supported.

Regression models with integer or float target field are not supported unless <Targets> element is specified with appropriate castInteger attribute.

Regression models with the <MiningSchema> element containing a reference to a <DerivedField> element are not supported.

Model Outputs

By default the target field will be available as an output field - this is a synonym for the predictedValue feature. Float target fields will always be cast to integer.

Supported Model Output Features	Description
predictedValue	The numeric dependent variable that we are predicting.
transformedValue	A value generated via a transformation expression applied to the predicted model output.
decision	A value generated via an expression applied to the predicted model output resulting in a categorized value.

Regression Classifier

A **PMML** on page 164 regression classifier combines the output from multiple regression equations to predict a categorical value.

Model Element

```
<RegressionModel functionName="classification" ...
```

Unsupported Features

Regression models with the <MiningSchema> element containing a reference to a <DerivedField> element are not supported.

Model Outputs

By default the target field will be available as an output field - this is a synonym for the predictedValue feature.

Supported Model Output Features	Description
predictedValue	The categorical dependent variable that we are predicting membership of.
transformedValue	A value generated via a transformation expression applied to the predicted model output.
decision	A value generated via an expression applied to the predicted model output resulting in a categorized value.
predictedDisplayValue	The human readable value used to represent the predicted value from the model.
probability	The statistical probability of the predicted value. Multiple probability outputs can be specified in the model, one for each predicted category or by rank.
residual	The residual of the probability output value (1 - probability) for the predicted value. Multiple residual outputs can be specified in the model, one for each predicted category or by rank.

Scorecard

A **PMML** on page 164 scorecard model is a regression based modeling technique mainly used to calculate risk or default probability.

Model Element

```
<Scorecard functionName="regression" ...
```

Unsupported Features

Scorecard models with the <MiningSchema> element containing a reference to a <DerivedField> element are not supported.

Model Outputs

By default the target field will be available as an output field - this is a synonym for the predictedValue feature and will always be of type 'double'.

Supported Model Output Features	Description
predictedValue	The calculated score.
transformedValue	A value generated via a transformation expression applied to the predicted model output.
decision	A value generated via an expression applied to the predicted model output resulting in a categorized value.
reasonCode	If specified the reason codes associated with the characteristics used to determine the score in order of their contribution to the final score. The rank of the request reason code can be specified otherwise the highest ranked reason code will be returned by default.

Read from Miner Dataset

Introduction to the Read from Miner Dataset

The Read from Miner Dataset stage is capable of extracting field information and data from a Miner dataset.

Reading from a Miner Dataset

This procedure describes how to read field information and data from a Miner Dataset.

1. Under **Primary Stages / Deployed Stages / Analytics Scoring**, drag the **Read from Miner Dataset** stage onto the canvas and connect it to the dataflow.

2. Double click the **Read from Miner Dataset** stage to show the **Read from Miner Dataset Options** dialog.
3. Click the ... button inside the **File Name** field to show the **Open File** dialog.
4. Using the **Open File** dialog, locate and select the focus file containing the Miner Dataset. Focus files have .ftr extension.
5. Click the **Fields** tab in the **Read from Miner Dataset Options** dialog to show the fields table. The fields table will be empty when configuring the **Read from Miner Dataset** stage for the first time and anytime a new focus file is selected (step 4 on page 174).
6. Click the **Regenerate** button. The fields table will list all the fields that the **Read from Miner Dataset** stage will read from the Miner Dataset.

Note: The **Regenerate** button will be disabled if no focus has been selected (steps 3 on page 174 and 4 on page 174).

Note: If the stage fails to retrieve fields from the Miner dataset then an error dialog will be displayed. Select a valid focus file and try again.

7. If required, modify the fields list using the available options (see **Fields Tab** on page 174).
8. Finally, click the **OK** button to apply the configuration.

If any validation errors are detected then a validation error message will be displayed. Correct the validation errors before clicking the **OK** button again. If no validation errors are found, the **Read from Miner Dataset Options** dialog will close.

Note: To re-configure the stage, double click the **Read from Miner Dataset** stage to re-display the **Read from Miner Dataset Options** dialog.

Fields Tab

The **Fields** tab contains a table listing all the fields that the **Read from Miner Dataset** stage will read from the Miner Dataset. The following table describes the options on the field that can be used to modify the table contents.

Option Name	Description
Regenerate	Replaces the fields currently defined with those read from the Miner dataset.

The following table describes the properties listed for all fields.

Column Name	Description	Editable
Miner Field Name	The name of the field as specified in the Miner Dataset.	No
Spectrum Field Name	The name of the field that will be created on the dataflow channel. If the Miner Field Name is not compatible with the Spectrum Technology Platform naming rules then the Spectrum Field Name will be different. Edit the text within the cell to specify the desired field name for the dataflow channel. Spectrum Technology Platform field names must: <ul style="list-style-type: none"> • Be unique. • Be non-empty. • Contain only alphanumeric, period or underscore characters. • Cannot start with a period. 	Yes
Spectrum Field Type	The data type of the field that will be created on the dataflow channel (see Output on page 175).	No
Include	A check box to specify whether to include this field in the dataflow channel.	Yes

Note: By clicking the table column headers the fields can be sorted by **Miner Field Name**, **Spectrum Field Name** or **Spectrum Field Type**.

Output

The following type mapping between Miner Field Type and Spectrum Field Type is done automatically

Table 19: Field Mappings

Miner Type	Spectrum Type
integer	integer
string	string
real	double
date	datetime

If a field has a name that contains symbols which are invalid in Spectrum, the invalid symbols will automatically be replaced with an `_`. If two or more fields have the same transformed name an additional index will be appended to the name, for example:

```
{x}Field{x}
{y}Field{y}
where {x} and {y} are characters not supported in a Spectrum field name, would
end up as
_Field_1
_Field_2
in the output channel.
```

Write to Miner Dataset

Introduction to the Write to Miner Dataset

The Write to Miner Dataset stage is capable of creating a Miner Dataset to hold the specified fields and data from a Spectrum dataflow.

Writing to a Miner Dataset

This procedure describes how to write field information and data to a Miner Dataset.

1. Under **Primary Stages > Deployed Stages > Analytics Scoring**, drag the **Write to Miner Dataset** stage onto the canvas and connect it to the dataflow.
2. Double click the **Write to Miner Dataset** stage to show the **Write to Miner Dataset Options** dialog.
3. Click the ... button inside the **File Name** field to show the **Save File** dialog.
4. Using the **Save File** dialog, choose where the new focus file will be saved and provide an appropriate file name.

Focus files must have `.ftr` extension. The `.ftr` extension will automatically be appended to the file name if it's missing.

5. Optional: Uncheck the **Overwrite** check box.

If a focus file with the same name already exists in the directory specified (step 4 on page 176), unchecking the **Overwrite** check box will prevent the existing focus file from being overwritten each time the stage is executed within a dataflow (the stage will fail to execute). However, this will mean that when the **Write To Miner Dataset** stage has been executed successfully within dataflow, the file path must be re-configured (step 4 on page 176) or the existing focus file will have to be manually moved or deleted.
6. Optional: Apply a metadata file for the new focus (see [Applying Metadata](#) on page 177).
7. Click the **Fields** tab in the **Write to Miner Dataset Options** dialog to show the fields table.

The fields table will list all the fields that the **Write to Miner Dataset** stage will write to the Miner Dataset.

8. Optional: Click the **Quick Add** button.
Clicking the **Quick Add** button launches the **Quick Add** dialog, the dialog lists all fields from the input channel and the current fields list. Select/deselect the checkboxes beside each field to add/remove the field from the fields list.
9. If required, modify the fields list using the available options (see **Fields Tab** on page 178).
At least one field must be specified or a validation error indicator will be displayed.
10. Finally, click the **OK** button to apply the configuration.
If any validation errors are detected then a validation error message will be displayed. Correct the validation errors before clicking the **OK** button again. If no validation errors are found, the **Write to Miner Dataset Options** dialog will close.

Note: To re-configure the stage, double click the **Write to Miner Dataset** stage to re-display the **Write to Miner Dataset Options** dialog.

Applying Metadata

A `.qsfm` metadata file contains additional information that can be applied to a focus file such as derived field definitions, field interpretations, binnings, record selections, field and focus comments and focus history. This procedure describes how to apply this kind of metadata to a focus file produced using the **Write to Miner Dataset** stage.

1. Obtain a valid Miner focus metadata file (either export from an existing focus in Spectrum Miner or acquire from a third party).
Metadata files must have a `.qsfm` extension.
2. Ensure that the **Write to Miner Dataset Options** dialog is currently displayed (see **Writing to a Miner Dataset** on page 176).
3. From the **Write to Miner Dataset Options** dialog, check the **Apply Metadata** check box.
4. Below the **Apply Metadata** check box, click the ... button inside the **File Name** field to show the **Open File** dialog.
The metadata file picker will be disabled when the **Apply Metadata** check box is not checked.
5. Using the **Open File** dialog, locate and open the `.qsfm` file obtained in step 1 on page 177.

Note: When the **Apply Metadata** check box is checked, a metadata file must be selected before the stage's configuration can be applied.

Note: Unchecking the **Apply Metadata** check box after selecting a metadata file will disable the metadata file chooser and will not apply the metadata file on stage execution, but for convenience the file chooser will not remove the selected file until the **Write to Miner Dataset Options** dialog is closed.

6. Optional: Check the **Ignore warnings** check box if you do not want the dataflow to fail if a piece of metadata cannot be applied.

Fields Tab

The **Fields** tab contains a table listing all the fields that the **Write to Miner Dataset** stage will write to the Miner Dataset. The following table describes the options on the field that can be used to modify the table contents.

Option Name	Description
Add	Clicking the Add button launches the Add dialog for adding a new field to the fields list. The Add dialog requires the new field's name, Miner type and length (if Miner type is 'string') to be specified.
Remove	Deletes the currently selected fields from the fields list.
Regenerate	Clicking the Regenerate button launches a dialog prompting for confirmation of the field regeneration. If the action is confirmed all existing fields are replaced with the fields found in the selected focus file. <p>Note: The Regenerate button will be disabled when there is no focus file selected.</p> <p>Note: Regenerating from an invalid focus will not clear or change the existing fields list.</p>
Quick Add	Clicking the Quick Add button launches the Quick Add dialog, the dialog lists all fields from the input channel and the current fields list. Select/deselect the check boxes beside each field to add/remove the field from the fields list. <p>Note: The Quick Add button will be disabled when a field's Miner field name is invalid.</p>

The field properties can also be updated by modifying cells in the following table columns directly.

Column Name	Description
Miner Field Name	<p>The name of the field as it will appear in the new Miner dataset. Edit the text within the cell to specify the desired field name. Field names must:</p> <ul style="list-style-type: none"> • Be unique. • Be non-empty. • Start with a letter. • Contain only alphanumeric or underscore characters. • Not be longer than 128 characters. <p>If a field name is changed to an invalid name then a validation error symbol will be displayed beside the fields table. Correct the validation error before clicking the OK button.</p>
Miner Field Type	<p>The Miner data type the field will use in the new Miner dataset (see Output on page 179). Change the field's data type by selecting from the drop down list.</p>
String Length	<p>If the chosen Miner field type in the table row is 'string' then the user will be able to specify an integer value between 1 and 4000 to set the length for the string field in the Miner focus. Any strings values for the field that are longer than the specified string length will be truncated to this length in the resulting focus.</p>

Output

The following type mapping between Spectrum Field Type and Miner Field Type is done automatically

Table 20: Field Mappings

Spectrum Type	Miner Type
integer	integer
string	string
double	real
datetime	date

All other Spectrum types map to Miner type 'String'.

Context Graph stages

Delete from Model

The Delete from Model stage deletes entities and relationships from a Context Graph model.

A dataflow that uses a Delete from Model stage requires an input stage that contains data from or queries the same model whose elements you are deleting. It also has two optional output ports: one contains data for the deleted entities and relationships and the other contains data for the records that were not deleted.

To configure a Delete from Model stage, you need to select the model you want to modify and then complete the Options tab and possibly the Runtime tab, depending on which write mode you want to use.

Input

The Delete from Model stage requires that your dataflow include an input stage that contains data from or queries the same model whose elements you are deleting. This requirement could be met by a Read from Model stage, a Query Model stage, a source stage of some kind, or even a control stage that combines multiple other input stages.

Options

The Options Tab

The Options tab enables you to configure the elements that will be removed from your model.

1. Select the name of the model whose elements you want to remove in the **Model** field.
2. Check **Delete entities** if you want to remove entities from the model.
 - Click the **ID** drop-down and select the field that contains the entities you want to remove from your model. This is often, but not always, the `_stp_id` field if the ID and type are together, or the `_stp_label` field if the ID and type are separate.
 - If the input data stores the entity type with the ID, such as with `_stp_id`, check the **In ID field** box.
 - If the input data stores the entity type separately, such as with `_stp_label`, check the **In separate field** box and select the appropriate field in the **Type** drop-down. This is often, but not always, the `_stp_type` field.
 - To select a specific entity type, check the **Literal** box and select the appropriate field in the **Type** drop-down.
3. Check **Delete relationships** if you want to remove relationships from the model.
 - Select the field that contains the source entities in the **Source ID** field. This is often `_stp_id` or `_stp_label`.

- If the input data stores the source entity type with the ID, such as with `_stp_id`, check the **In ID field** box.
 - If the input data stores the source entity type separately, such as with `_stp_label`, check the **In separate field** box and select the appropriate field in the **Type** drop-down. This is often, but not always, the `_stp_type` field.
 - To select a specific source entity type, check the **Literal** box and select the appropriate field in the **Type** drop-down.
 - Select the field that contains the relationship between the source and target entities in the **Label** drop-down.
 - If there are multiple relationships with the same label between the source and target entities, click the **Unique ID in separate field** box and select the name of the **Unique ID**.
 - Select the field that contains the target entities in the **Target ID** field. This is often `_stp_id` or `_stp_label`.
 - If the input data stores the target entity type with the ID, such as with `_stp_id`, check the **In ID field** box.
 - If the input data stores the target entity type separately, such as with `_stp_label`, check the **In separate field** box and select the appropriate field in the **Type** drop-down. This is often, but not always, the `_stp_type` field.
 - To select a specific target entity type, check the **Literal** box and select the appropriate field in the **Type** drop-down.
4. Click **OK**.

The Runtime Tab

The Runtime tab allows you to control processing options.

1. Click **Concurrent writes** if you want to allow the model to be written to by multiple Context Graph stages at the same time. Click **Exclusive lock** (default) if you do not want to allow the model to be written to by multiple Context Graph stages. When this mode is checked, properties can be updated after they are created.
2. Click **OK**.

Output

The Delete from Model stage has two optional outgoing ports to which you can attach various sink stages. One sink captures data for the successfully deleted entities and relationships, while the other collects any records that the dataflow did not process correctly. This is called the Error Port, and records that pass through this port into the sink are considered malformed.

Capturing malformed records can help you identify the problem with those records. When you attach a sink to the Error Port, the resulting output file will contain all the fields from the malformed records. It will also contain a Reason field that specifies why the record failed.

Import to Model

The Import to Model stage populates data in a new or existing Context Graph model from two incoming channels.

The Import to Model stage is used to create a complex network of relationships called a Context Graph model, which can be displayed in a model. It can also be used to populate an existing model. Once a model is created, it can be queried against in a Read from Model stage or a Query Model stage, and it can be visualized in Discovery or the Relationship Analysis Client.

A dataflow that uses an Import to Model stage requires two incoming channels of data: one for entities and one for relationships. You'll then need to complete the Entities tab and the Relationships tab in the Import to Model stage to complete your dataflow and create or update your model.

Input

The Import to Model stage requires that your dataflow contain two channels: one that provides data for entities going into the Entity Port (the top port) and one that provides data for relationships going into the Relationship port (the bottom port). This requirement could be met by two source stages (each containing one input file), or it could come from multiple source stages that feed into Record Combiners and ultimately become two streams, or it could come from one source file whose data goes through a Conditional Router or a Splitter that outputs the data into two streams. It doesn't matter which method you use as long as the end result is a channel of entity data and a channel of relationship data that go into the Import to Model stage.

Entity Data

Data going into the Entity Port needs to include both type and ID information for your entities. You can have a Type field ("Person") and an ID field ("Bob"), or you can have just an ID field that combines both type and ID information, separated by a colon ("Person:Bob"). For instance, your file could look something like the comma-delimited data below. The Type field tells us that the entities are people and places, and the ID field provides the names of the people and places.

```
Type, ID
Person, Robert
Person, James
Person, John
Place, London
Place, New York
Place, Los Angeles
Place, San Francisco
```

Alternatively, your input file could contain a single field that combines both type and ID:

```
ID
Person: Robert
Person: James
Person: John
Place: London
Place: New York
Place: Los Angeles
Place: San Francisco
```

Note: The fields that contain type and ID data do not actually need to be named "Type" and "ID"; any field name is acceptable.

Relationship Data

Data going into the Relationship Port needs to include fields that identify source types, source IDs, target types, target IDs, and labels that identify the relationships between the sources and targets. Note that all source and target entity information must reference entities that are provided on the Entity Port. Your relationship data may also include properties about those relationships. For instance, your file could look something like the data below. In this case, the `SourceType` field tells us that all sources are people, and the `TargetType` field tells us that the targets are people and places. The `SourceID` field provides names of all the sources, and the `TargetID` field provides names of the people and places. The `Label` field identifies the relationships, in this case "Works_With", "Works_At", or "Lives_At".

```

SourceType, SourceID, Label, TargetType, TargetID
Person, Robert, Works_With, Person, James
Person, Robert, Works_With, Person, John
Person, James, Works_With, Person, Robert
Person, James, Works_With, Person, John
Person, John, Works_With, Person, Robert
Person, John, Works_With, Person, James
Person, Robert, Works_At, Place, London
Person, James, Works_At, Place, London
Person, John, Works_At, Place, London
Person, Robert, Lives_At, Place, New York
Person, James, Lives_At, Place, Los Angeles
Person, John, Lives_At, Place, San Francisco

```

Options

The Entities Tab

The Entities tab enables you to configure the entities that will be included in your model. These entities represent objects or events, which may have properties associated with them, and these properties can be stored in your model as well if you choose to include it. Entities are linked to each other via relationships, which you will establish on the Relationships tab.

1. Enter the name of your model in the **Model** field.
2. Click the **ID** drop-down and select the field whose data you want to use to generate the entities for your model.
3. If the entity's type is contained in its own field, check the **Type in separate field** box and select the appropriate field in the **Type** drop-down.
4. The **Internal Index** grid includes a list of fields that are generated by the Import to Model stage. The `_stp_id` field is always indexed; the `_stp_label` and `_stp_type` fields are optional. These fields can be indexed with or without case sensitivity.
5. The **Field Name** grid includes all the fields from your entity input file. Select the fields whose data you want included in the model by clicking the **Include** box for those fields.

6. Select which fields you want to be indexed in your model by clicking the **Index** box for those fields. Selecting which fields to index, rather than indexing all fields in your model, results in faster performance when writing to a model. However, if you later attempt to query fields in your model that were not indexed, the response time will be slower. For example, the **Specify starting entity** option in the query tool for the Relationship Analysis Client works only on indexed properties. You can query non-indexed properties using conditions, but the performance will be slower.
7. In the **Index Type** column, you can choose whether the data should be indexed with or without case sensitivity. Selecting Case Insensitivity typically results in greater response to a search. You cannot change the Index Type for an existing property unless it has zero counts within the model. In other words, if your model contains a property but none of the records that make up the model uses that property, you can change the index type. If one or more records uses the property, you cannot change the index type.

Note: The `_stp_id`, `_stp_type`, and `_stp_label` properties are internal properties and will always appear in the list of indexed fields. You can deselect `_stp_type` and `_stp_label`, but `_stp_id` must be indexed; however, you are able to designate whether its index type should be exact or with case insensitivity.
8. Click the **Relationships** tab to continue creating your model.

The Relationships Tab

After determining the entities for your model, you need to establish the relationships between source and target entities on the Relationships tab. These relationships represent the connection between two entities (for example, John Smith is a customer of ABC Enterprises, Inc.). As with entities, relationships may also contain properties, which you may or may not choose to include in your model.

1. Select the field that contains the source entity ID in the **Source ID** field.
2. If the source entity type is contained in its own field, check the **Type in separate field** box and select the appropriate field in the **Type** drop-down.
3. Select the field that contains the relationship between the source and target in the **Label** drop-down.
4. If you want to allow a relationship to be created more than once between a source and target entity, click the **Allow more than one relationship based on unique ID** box and select the field on which to base the relationship in the **Unique ID** drop-down.
5. Select the field that contains the target entity ID in the **Target ID** field.
6. If the target entity type is contained in its own field, check the **Type in separate field** box and select the appropriate field in the **Type** drop-down.
7. The **Field Name** grid includes all the fields from your entity input file. Select the fields whose data you want included in the model by clicking the **Include** box for those fields.
8. Click **OK**.

The Options Tab

The **Update existing model** field designates whether to update, append, remove, or retain data when an Import to Model job is run and the model already exists. If this box is unchecked and you run a job that writes new data to the model, the existing data will first be deleted; if the new job results in blanks for a property, that property's existing data will be removed.

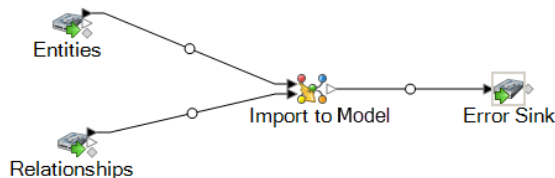
Note: Import to Model has no error recovery capability; if you use this option, be sure to back up your existing model before updating the model with this option selected.

If this box is checked, you must specify how Import to Model should handle existing data for both entities and relationships:

- Always update properties—Overwrite existing data with input data regardless of contents. If the input properties are empty or if properties are in the existing data but not the input data, those properties will be removed from the record.
- Update properties unless all input is null—Overwrite existing data with input data unless **all** of the input properties are empty, in which case the record will be written to the error port.
- Never overwrite non-empty properties—Do not overwrite or remove existing data with input data unless the existing properties are empty. Input data will be appended to the record.
- Never overwrite properties with empty input data—Overwrite existing data with input data but do not remove existing properties when input properties are empty.

Output

The Import to Model stage has an optional outgoing port to which you can attach a sink stage that collects any records that the dataflow did not process correctly. This is called the Error Port, and records that pass through this port into the sink are considered malformed.



Capturing malformed records can help you identify the problem with those records. When you attach a sink to the Error Port, the resulting output file will contain a superset of the fields from both input files. It will also contain a Reason field that specifies why the record failed. So, for example, if your entities input file contains Type, ID, and Location fields, and your relationships input file contains Type, ID, and Label fields, your output file would contain Reason, Type, ID, Location, and Label fields.

Causes for record failure include, but are not limited to, the following:

- In the relationship configuration, the source entity equals the target entity.
- Relationships reference an entity that has not been defined.
- Duplicate entities or relationships exist.
- Input fields are out of order.

- Type, ID, or label fields are empty.

Merge Entities

The Merge Entities stage merges two or more entities into a single entity.

A dataflow that uses a Merge Entities stage requires one or more inputs that identify the entities you are merging. It also has two optional output ports: one contains data for the merged entity and its properties and the other contains data for the records that could not be merged.

To configure a Merge Entities stage, you need to select the model you want to modify and then complete the **Options** tab and possibly the **Runtime** tab, depending on which write mode you want to use.

Input

The Merge Entities stage requires that your dataflow include an input stage that contains data from or queries the same model whose elements you are merging. This requirement could be met by a Read from Model stage, a Query Model stage, a source stage of some kind, or even a control stage that combines multiple other input stages.

Options

The Options Tab

The Options tab enables you to configure the elements that will be merged in your model.

1. Select the model whose entities you want to merge in the **Model** drop-down list box.
2. Select or enter the type of entities you are merging in the **Type** drop-down list box.
3. Select **Merge multiple sources on a single record** if you want to spread the source information across multiple fields in a single record. Select **Merge a single source on multiple records** if you want the source information to be in a single field across multiple records.
4. If you are merging multiple sources on a single record:
 - a. Select the field that contains the source entities in the **ID** field. This is often, but not always, the `_stp_id` field if the ID and type are together, or the `_stp_label` field if the ID and type are separate.
 - b. If the input data stores the entity type with the ID, such as with `_stp_id`, check the **In ID field** box.
 - c. If the input data stores the entity type separately, such as with `_stp_label`, check the **In separate field** box and select the appropriate field in the **Type** drop-down list box. This is often, but not always, the `_stp_type` field.
 - d. To select a specific entity type, check the **Literal** box and select the appropriate field in the **Type** drop-down list box.
 - e. Repeat steps a and b to define at least one more source ID for the merge. You can add as many sources as you like. You can also delete any source by clicking the red "X" that is to the right of the source ID.

Note: By default, the first source will be considered the master source, but you can designate any of the sources to be the master. Properties for the master source will not be overwritten by properties for the other sources.

5. If you are merging a single source on multiple records:
 - a. Select the appropriate field in the **Group by** drop-down list box.
 - b. Select the field that contains the source entities in the **ID** field.
 - c. If the input data stores the entity type with the ID, such as with `_stp_id`, check the **In ID field** box.
 - d. If the input data stores the entity type separately, such as with `_stp_label`, check the **In separate field** box and select the appropriate field in the **Type** drop-down. This is often, but not always, the `_stp_type` field.
 - e. To select a specific entity type, check the **Literal** box and select the appropriate field in the **Type** drop-down list box.

Note: The entity in the first record of each group is considered the master. Properties for the master source will not be overwritten by properties of other records.

6. Click **OK**.

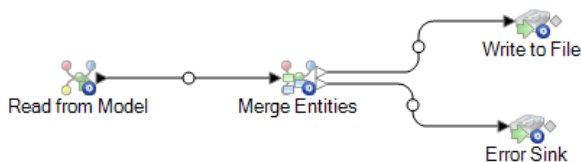
The Runtime Tab

The Runtime tab allows you to control processing options.

1. Click **Concurrent writes** if you want to allow the model to be written to by multiple Context Graph stages at the same time. Click **Exclusive lock** (default) if you do not want to allow the model to be written to by multiple Context Graph stages. When this mode is checked, properties can be updated after they are created.
2. Click **OK**.

Output

The Merge Entities stage has two optional outgoing ports to which you can attach various sink stages. One sink captures data for the successfully merged entity and its properties, while the other is used to collect data for the records that could not be merged. This is called the Error Port, and records that pass through this port into the sink are considered malformed.



Capturing malformed records can help you identify the problem with those records. When you attach a sink to the Error Port, the resulting output file will contain a superset of the fields from all records. It will also contain a Reason field that specifies why the record failed. So, for example, if one record

contains Type, ID, and Location fields, and a second record contains Type, ID, and Label fields, your output file would contain Reason, Type, ID, Location, and Label fields.

Causes for record failure include, but are not limited to, the following:

- The ID field value is empty.
- The ID entered does not return any entities from the selected model.
- The ID and Type combination entered does not return any entities from the selected model.
- The ID and Type combination does not use a valid format (the required format is TYPE_VALUE:ID_VALUE).

Read from Model

The Read From Model stage executes a query to read data from an existing model.

A source stage that uses a saved or new query to read the data inside an existing Context Graph model. It then returns that data as fields in the dataflow output stage and makes it available for use with other stages or processes.

The Query Tab

The Query tab allows you to provide a query that returns data from the model you select. The model data is returned as data rows in the dataflow output. You can use the query builder provided, or you can create a custom query.

You can also select from or modify existing queries that are listed in the Query drop-down. If you modify existing queries and want to save those changes but also retain the original query, be sure to save the modified query under a new name; otherwise, you will overwrite the existing query. If you apply a different query to the model or cancel out of the application, changes to the query will be lost.

Follow these instructions to use the query builder:

1. Choose whether you want to **Select elements** included in the query results, **Add elements** included in the query results, or **Show result** by highlighting the results on the canvas.
2. Check the **Include results from partial traversals** box to include the results from each step in the query. Leave the box unchecked to include only the results that meet the requirements of the last step. For example, let's say that you are looking at a model that depicts world-wide terrorist activity for the events leading up to September 11, 2001, and you want to return data for any meetings that both Osama bin Laden and Mohamed Atta attended. Your query might include the following steps:
 - An initial step that includes an exact search type for entities with an `_stp_id` property that has a literal value of "Person:Osama bin Laden"
 - An Entity to Relationship step of connected with a relationship label of "Meeting"
 - A Relationship to Entity step of connected with a condition of an `_stp_id` property that contains "Atta"

If you leave the **Include results from partial traversals** box unchecked, the query would return a single meeting between Osama bin Laden and Mohamed Atta. If you check the box, the query would return all meetings between Osama bin Laden and anyone else in the model. The additional

records would be returned because the second step looks for meetings attended by Osama bin Laden and checking the box returns results from each step in the query.

3. Leave the **Include results from circular traversals** box checked to include elements that occur more than once in each traversal. Uncheck the box to include those elements just once in each traversal. For example, let's say you are using the same model mentioned in step 1, and you initially want to return data for any meetings that Mohamed Atta attended but once you have those results, you want to see all attendees of a particular meeting. Your query might include the following steps:
 - An initial step that includes an exact search type for entities with an `_stp_id` property that has a literal value of `Person:Mohamed Atta`, which will return his entity
 - An Entity to Relationship step of connected with a relationship label of "Attended", which will return all events that he attended, including a meeting in Kandahar
 - A Relationship to Entity step of connected with a condition of an Event property that contains "Kandahar", which will return just the Kandahar meeting
 - An Entity to Relationship step of connected with a relationship label of "Attended", which will return relationships that connect to three other entities who attended the meeting in Kandahar and may or may not return the (already traversed) relationship that connects to Mohamed Atta, depending on whether you use this option.

If you leave the **Include results from circular traversals** box checked, Mohamed Atta's relationship will be returned in addition to those for the other three attendees. If you uncheck the box, Mohamed Atta's relationship will not be returned because that relationship (Person:Mohamed Atta->Attended->Meeting:Kandahar) was already traversed in the first step of the query.

4. Leave the **Limit results to** box checked and enter a number to specify the total maximum number of entities and relationships to return from the query. The default is 5000. The number entered here applies to unique elements, so if the same element appears in multiple results, they will count as one result. To avoid this scenario, use the dedup function discussed at the end of this topic; it will remove duplicate results in the output. If your root step returns a list and you are querying a large model, we strongly suggest entering a limit in this field to prevent the server from becoming unresponsive.

Note: Limits can be set here or as Query Result Limits set in Relationship Analysis Client **General Settings**; if the limits are not the same, the lower limit will prevail.

5. Complete the Selection tab.
 - Click **All entities**, **All relationships**, or **Specify starting entities** to identify what you want to query against. The Specify starting entities selection allows you to determine at what point in the model you want to begin your search. For instance, if you are looking at a model that depicts world-wide terrorist activity during specific years, you might have country names for entities. Rather than query against the entire model, you might want to look at activity just in Afghanistan. In this case you could select "All" as the **Search type** and "Country" as the **Property name**, leave **Literal** selected, and enter "Afghanistan" as the **Property value**.

Note: The value in the **Property value** field is case sensitive if that field was indexed with the Exact type selected (versus Case insensitive). For more information on selecting types when indexing fields, see [The Entities Tab](#) on page 211.

You could also click **Field** and select "Location" as the Property value, for example, rather than entering a specific value. If you select Field, an **Input Data** grid containing the name of the field you just selected will appear under the query name along with a cell where you can enter the default value. If you reuse this query elsewhere, you can use the default value you provided in this step, or you can override the default at that time.

- If you clicked **All entities** or **Specify starting entities**, select the **Entity types** for your query. You can choose to query selected types or all types. Click **Select None** to deselect any selected types. In addition to returning a more focused set of results, selecting entity types will affect other factors such as which properties and fields are available in the first step of the query, which directions, entity types, and relationship labels are available in subsequent steps of the query, and so on.
- If you clicked **Specify starting entities**, select the **Search type**:

Exact	Searches the index for data that matches exactly what you enter on the Selection tab, including casing. As with property values, the value here is case sensitive if that field was indexed with the Exact type selected; if case sensitivity was used and you search for "texas" while your data includes entries of "Texas", they will not be returned.
Starts with	Searches the index for data that contains text beginning with what you enter on the Selection tab. The search does not need to be a complete word. For example, a literal property value of "tech" or "tec" would be considered a match for a property value containing "Technical", "Technology", "Technologies", or "Technician".
Ends with	Searches the index for data that contains text ending with what you enter on the Selection tab. The search does not need to be a complete word. For example, a literal property value of "Emirates" or "tes" would be considered a match for a property value containing "United Arab Emirates".
Contains	Searches the index for data that contains the text that you enter on the Selection tab. The search string can incorporate any portion of an expected match. For example, a literal property value of "Light" and "Light Amplification" would be considered a match for a property value containing "Light Amplification by Stimulated Emission of Radiation".

Any	Searches the index for data that contains any of the text that you enter on the Selection tab. For example, a literal property value of "Austin Tex" would be considered a match for a property value containing "Texarkana" or "Stephen F. Austin University".
All	Searches the index for data that contains all of the text that you enter on the Selection tab. For example, a literal property value of "Allstate claim 2013" would be considered a match for a property value containing "filed claim with Allstate June 2013", as would literal property values of "all state" or "all 13".
Between	<p>Searches the index for data that falls within a range that you specify on the Selection tab. When you use this search type, you must select a Property name that contains date, time, date/time, or numeric data. All numeric data types are supported with the exception of BigDecimal. For example, the following specifications would return all entities with StartDate values occurring in the year 2000:</p> <ul style="list-style-type: none">• A Property name of "StartDate" that is a Date type• A literal Start value of "1/1/2000"• A literal End value of "12/31/2000"
Fuzzy	Searches the index for the text you enter on the Selection tab but allows for some differentiation (missing letters, extra letters, or substitutions of letters). The amount of differentiation that is acceptable to still be considered a match depends on what you enter in the Metric field. This figure must be greater than zero and less than one; in other words, it must range from ".1" to ".9". For example, if you search for "Barton" and enter ".9" as the metric, the search will return records with "Carton" (replaces B with C), "Bartons" (adds s), and "Baton" (removes r), because all of these words are one character different from the search word "Barton".

Wildcard

Searches the index for the text you enter on the Selection tab but allows for a single wildcard character or a wildcard character sequence. Supported wildcard characters include the question mark (?), which matches any single character, and the asterisk (*), which matches any character sequence (including blanks). For example, if you search cities in Texas for "Aus*", the search will return records with "Austin", "Austonio" and "Austwell". If you conduct a similar search for "Aust??", only "Austin" will be returned because each question mark represents a single character and the other two city names have more characters in their name.

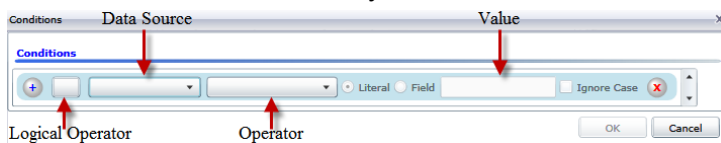
Note: A query that includes an asterisk wildcard as the first character in the search string may result in a lengthy response time.

- If you clicked **Specify starting entities**, select the **Property name** from the drop-down list. This list contains all properties associated with the entities and relationships that make up the model.

Note: You can only query properties that have been indexed; non-indexed properties will not appear in the Property name drop-down.

- If you clicked **Specify starting entities**, and selected a search type other than "Between", select the **Property value**. You can click **Literal** and enter a text string to be used in the search. Alternatively, you can click **Field** and select the field whose data should be searched; if you choose this option you will also need to enter a value in the Input Data grid.
- If you clicked **Specify starting entities**, and selected the "Between" search type, select the **Start value** and **End value** to enter the range. You can click **Literal** and enter a value to be used in the search. Alternatively, you can click **Field** and select the field whose data should be searched; if you choose this option you will also need to enter a value in the Input Data grid.

6. Complete the **Conditions** tab if you want to place additional constraints on the query. The Conditions tab has four entry fields:



- a. If you are creating the first condition, the **Logical operator** field will remain empty. If you are creating a subsequent condition, specify whether this condition should be used in conjunction with previous conditions ("And") or if it should be used instead of previous conditions ("Or").
- b. Select the property on which the condition will be based in **Data source** field.
- c. Select an operator for the condition that is appropriate for the data type in the **Operator** field:

Equals	Searches model elements for properties with values that match exactly what you enter in the Value field. This can be a numeric value or a text value.
Not Equals	Searches model elements for properties with values that have any value other than what you enter in the Value field. This can be a numeric value or a text value.
Exists	Searches the model elements for the existence of the property that you select in the Data Source field.
Does not Exist	Searches the model for elements that do not contain properties that you select in the Data Source field
Is Blank	Searches model elements for properties that contain no data. If a property value is blank, that element will be returned. This can be a numeric value or a text value.
Is Not Blank	Searches model elements for properties that contain any data. If a property value is not blank, that element will be returned. This can be a numeric value or a text value.
Greater Than	Searches model elements for properties whose values are greater than the value you specify. This can be a numeric, date, date/time, or time value.
Greater Than or Equals	Searches model elements for properties with numeric values that are greater than or equal to the value you specify. This can be a numeric, date, date/time, or time value.
Less Than	Searches model elements for properties with numeric values that are less than the value you specify. This can be a numeric, date, date/time, or time value.
Less Than or Equals	Searches model elements for properties with numeric values that are less than or equal to the value you specify. This can be a numeric, date, date/time, or time value.
Contains	Searches model elements for properties with values that contain what you enter in the Data Source field. The search does not need to be a complete word. For example, a literal property value of “Light” or “Light Amplification” would be considered a match for a property value containing “Light Amplification by Stimulated Emission of Radiation”, so it would be returned. This can be a numeric value or a text value.
Does not Contain	Searches model elements for properties that do not contain what you enter in the Data Source field. The search does not need to be a complete word. For example, a literal property value of “Light” or “Light Amplification” would be considered a match for a property value containing “Light Amplification by Stimulated Emission of Radiation”, so it would not be returned. This can be a numeric value or a text value.

Starts With	Searches model elements for properties whose values start with the text you enter in the Data Source field. For example, if you enter "Van" for the LastName field it would return results with "Van Buren", "Vandenburg", or "Van Dyck".
Does Not Start With	Searches model elements for properties whose values do not start with the text you enter in the Data Source field. For example, if you enter "Van" for the LastName field it would not return results with "Van Buren", "Vandenburg", or "Van Dyck" but would return results with "Eddie Van Halen".
Ends With	Searches model elements for properties whose values end with the text you enter in the Data Source field. For example, if you filter for records that end with "burg" in the City field, it would return results with "Gettysburg", "Fredricksburg", and "Blacksburg".
Does Not End With	Searches model elements for properties whose values do not end with the text you enter in the Data Source field. For example, if you filter for records that end with "burg" in the City field it would not return results with "Gettysburg", "Fredricksburg", and "Blacksburg" but would return results with "Burgess".
Match Regular Expression	Searches the model elements for properties having a regular expression match for what you enter in the Data Source field. Regular expression matches identify strings of text of interest, such as particular characters, words, or patterns of characters. The value field should contain a valid regular expression pattern.
Is Roughly Similar To	Searches model elements for properties with values close to what you enter in the Data Source field but allows for some differentiation (missing letters, extra letters, or substitutions of letters). This operator is equivalent to the Fuzzy search type with search metric of .5.
Is Similar To	Searches model elements for properties with values close to what you enter in the Data Source field but allows for some differentiation (missing letters, extra letters, or substitutions of letters). This operator is equivalent to the Fuzzy search type with search metric of .6.
Is Very Similar To	Searches model elements for properties with values close to what you enter in the Data Source field but allows for some differentiation (missing letters, extra letters, or substitutions of letters). This operator is equivalent to the Fuzzy search type with Search Metric of .7.

d. In the drop-down box following the list of operators:

- Select **Literal** and enter a text string the fourth box (called the **Value** field) to be used in the query.
- Select **Field** and then select the field whose data should be searched in the **Value** field.
- Select a previous step (such as "**Root**" or "**Step1**") and then a property in the **Value** field to compare property values for the current step against values returned in a previous step.

(Note that if you named the output on the Output tab of previous steps, those names will appear in the drop-down rather than "Root" or "Step1".) In this case, the properties shown in the Value field are based on properties for the previous step. For example, if you knew the name of one person (Mohamed Atta) who attended a particular event (a meeting in Kandahar) but wanted to know the names of the other attendees, you could create the following query that includes a property value comparison:

- A root step that looks for an entity type Person with an `_stp_id` of that contains "Mohamed"
- An Entity to Relationship step with a relationship label of "Attended"
- A Relationship to Entity step with a condition that includes an Event that contains "Kandahar"
- An Entity to Relationship step with a relationship label of "Attended" plus a condition that this step's `_stp_id` does not contain the same `_stp_id` value that was found in the root step.

This query will find that Mohamed Atta attended an event in Kandahar and that it was also attended by three other people whose `_stp_id` value is not "Mohamed".

- e. Click **Ignore Case** if the query results can be either upper or lower cased.
- f. Repeat steps a through e to add additional conditions.
- g. Click **OK**.

If, for example, you want to target terrorist activity in Afghanistan between 2001 and 2010, you would create two conditions. First, you would select "date" for the **Property name**, then "Greater Than or Equals", leave Literal selected, and then enter "2001". You would follow this with a second condition set to "And" that also uses "date," then "Less Than or Equals", then "2010". Alternatively, you could click **Field** and select "Date" rather than entering a specific value. Add, delete, or change the order of conditions by using the icons on either side of the conditions. Click **Ignore Case** if the query results can be either upper or lower cased.

7. Complete the **Output** tab to define how you want your output to appear.

- Click the **Include in results** box if you want the results from this step to be included in the output.

Note: This box must be checked for the last step in any series; therefore, if there is only one step you cannot uncheck this box.

- Click **Specify name** and enter text in the **Name** field to provide a name for this step in the output. Click **List** to use this entry as the name and type of the field in hierarchical output; leave it unchecked to have this entry added as a prefix for all output fields. Using the example from step 3, you might call this step "Afghanistan". Output fields from this step may be named "Afghanistan.Latitude" or "Afghanistan.Date".
- Click **Use type name** to use the field type as the name for this step in the output. Entities will use entity types and relationships will use relationship labels. Continuing with the same example, output fields with this selection may be named "Person.Latitude" or "Person.Date". If you select this option and enter a name in the **Name** field, that name will also be added as a prefix for all output fields in addition to the field type. Continuing with the same example, output fields with this selection may be named "Afghanistan.Person.Latitude" or "Afghanistan.Person.Date".

8. Specify the steps you want the query to take by selecting the appropriate option in the **Add Operations** drop-down. You can complete this step for the Flow, Conditions, or Output tab. Note that your options vary by whether the root element is an entity or a relationship.

- If you choose **Entity to Entity** (valid for All entities and Specify starting entities), you can then refine your search to return data based on relationship labels between two entities (Connected), before entities (Predecessors), or after entities (Successors). For example, if you are querying a model of family members, and you choose a Relationship label of "Father," a Connected query will return all entities that have a Father label between them (in other words, fathers, sons, and daughters). A Predecessors query will return all entities who are a source entity of a Father relationship connected to another entity (in other words, fathers). A Successors query will return all entities who are the target entity of a Father relationship connected to another entity (in other words, sons and daughters).

As in the root step of your query, you can also select Entity types for this step of the query. You can choose to query selected types or all types. Click Select None to deselect any selected types.

- If you choose **Entity to Relationship** (valid for All entities and Specify starting entities), your options are very similar to those for Entity to Entity. You can refine your search to return data based on relationship labels that attach two entities (Connected), occur before entities (Predecessors), or occur after entities (Successors). You can also add conditions to and define output for the query.
- If you choose **Relationship to Entity** (valid for All relationships), you can refine your search to return data based on conditions you set. You can return data when a condition is in place for an entity that is connected to another entity (Connected), an entity that is a source to a relationship (Predecessors), and for when an entity is a target of a relationship (Successors). As in the root step of your query, you can also select Entity types for this step of the query. You can choose to query selected types or all types. Click Select None to deselect any selected types.

Regardless of the type of operation you add, you can create Conditions for that operation. You can also define how you want the output from this step to appear. You will notice that steps subsequent to the root step are given a path for output. The path and the step name define the hierarchy of your output data. If you checked the List box in the root step, this path will default to being part of the path in the step before it; however, you can remove the name of the root step. For example, if you named the root step "Locations" and clicked the List box, the first step would by default show "/Locations" in the **Path** field. (Alternatively, you could remove "Locations" and leave just the slash to have this step originate at the root.) If you called the first step "CountryName", the second step would by default show "/Locations/CountryName" in the **Path** field and the Locations field output would contain a list of CountryName results. Click the **Include in results** box if you want the results from this step to be included in the output. Click **Dedup** if you want the query to remove duplicate results from the output.

9. Click **OK**.

The Fields Tab

The Fields tab allows you to designate fields to be returned in the output of your dataflow.

Note: The **Query** tab must be completed before the **Fields** tab is completed unless you are creating a custom query.

If you are using the query builder and have a field with a Dynamic Model Fields type, you can change the names of the fields that are returned for that step by selecting the step and clicking **Modify**. This will access the **Modify Field** dialog box, which contains a grid that is populated with fields and properties that map to the results of the step you selected. Similarly, the structure of this data is determined by whether you chose list output as well as the paths and names of the steps you entered on the Query tab.

If you are building a custom query, you can add, modify, or remove fields from your query.

Follow these steps to add fields to a custom query.

1. Click **Add** to open the **Add Field** dialog box.
2. Select the type of output field you want to add from the **Field type** drop-down box. The following data types are supported:

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 Management Console.
datetime	A data type that contains a month, day, year, and hours, minutes, and seconds. For example, 2012/01/30 6:15 PM.
double	A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is 4.9E-324 to 1.7976931348623157E308. For information on E notation, see en.wikipedia.org/wiki/Scientific_notation#E_notation .
float	A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is 1.4E-45 to 3.4028235E38. For information on E notation, see en.wikipedia.org/wiki/Scientific_notation#E_notation .
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.

If you want to add a field from your model, select **Dynamic Model Fields** and the tab will be populated with fields and properties from your model. The structure of this data is determined by whether you choose list output as well as the paths and names entered on **Add Field** dialog box.

You can also add a new, user-defined, custom data type if necessary. Among other types of data, a new type can be a list of any defined data type (though you could select most types from the drop-down options and click the List check box).

3. Enter the **Field name** for the field you want to query in the model.
4. Click **List** to return output in a hierarchical format; leave it unchecked to add prefixes to output fields.
5. Click **Cancel** when you're done adding fields.
6. Click **OK**.

Output

The Read From Model stage requires that your dataflow contain an output stage that has defined, at minimum, the field or fields that you are querying. Otherwise, that data will not appear in your output. For example, if your Read From Model stage queries the `_stp_id` field in your graph database, your output stage must contain a field that captures that data.

Query Model

The Query Model stage is an intermediate stage that uses incoming data rows to execute queries that extract specific entities and relationships from a model.

For example, Query Model can be used as part of a service to understand a customer's influence score within the network or determine if a customer record already exists in the graph database.

The Query Tab

The Query tab allows you to provide a query that returns data from the model you select. The model data is returned as data rows in the dataflow output. You can use the query builder provided, or you can create a custom query.

You can also select from or modify existing queries that are listed in the Query drop-down. If you modify existing queries and want to save those changes but also retain the original query, be sure to save the modified query under a new name; otherwise, you will overwrite the existing query. If you apply a different query to the model or cancel out of the application, changes to the query will be lost.

Follow these instructions to use the query builder:

1. Choose whether you want to **Select elements** included in the query results, **Add elements** included in the query results, or **Show result** by highlighting the results on the canvas.

2. Check the **Include results from partial traversals** box to include the results from each step in the query. Leave the box unchecked to include only the results that meet the requirements of the last step. For example, let's say that you are looking at a model that depicts world-wide terrorist activity for the events leading up to September 11, 2001, and you want to return data for any meetings that both Osama bin Laden and Mohamed Atta attended. Your query might include the following steps:

- An initial step that includes an exact search type for entities with an `_stp_id` property that has a literal value of "Person:Osama bin Laden"
- An Entity to Relationship step of connected with a relationship label of "Meeting"
- A Relationship to Entity step of connected with a condition of an `_stp_id` property that contains "Atta"

If you leave the **Include results from partial traversals** box unchecked, the query would return a single meeting between Osama bin Laden and Mohamed Atta. If you check the box, the query would return all meetings between Osama bin Laden and anyone else in the model. The additional records would be returned because the second step looks for meetings attended by Osama bin Laden and checking the box returns results from each step in the query.

3. Leave the **Include results from circular traversals** box checked to include elements that occur more than once in each traversal. Uncheck the box to include those elements just once in each traversal. For example, let's say you are using the same model mentioned in step 1, and you initially want to return data for any meetings that Mohamed Atta attended but once you have those results, you want to see all attendees of a particular meeting. Your query might include the following steps:

- An initial step that includes an exact search type for entities with an `_stp_id` property that has a literal value of Person:Mohamed Atta, which will return his entity
- An Entity to Relationship step of connected with a relationship label of "Attended", which will return all events that he attended, including a meeting in Kandahar
- A Relationship to Entity step of connected with a condition of an Event property that contains "Kandahar", which will return just the Kandahar meeting
- An Entity to Relationship step of connected with a relationship label of "Attended", which will return relationships that connect to three other entities who attended the meeting in Kandahar and may or may not return the (already traversed) relationship that connects to Mohamed Atta, depending on whether you use this option.

If you leave the **Include results from circular traversals** box checked, Mohamed Atta's relationship will be returned in addition to those for the other three attendees. If you uncheck the box, Mohamed Atta's relationship will not be returned because that relationship (Person:Mohamed Atta->Attended->Meeting:Kandahar) was already traversed in the first step of the query.

4. Leave the **Limit results to** box checked and enter a number to specify the total maximum number of entities and relationships to return from the query. The default is 5000. The number entered here applies to unique elements, so if the same element appears in multiple results, they will count as one result. To avoid this scenario, use the dedup function discussed at the end of this topic; it will remove duplicate results in the output. If your root step returns a list and you are querying a large model, we strongly suggest entering a limit in this field to prevent the server from becoming unresponsive.

Note: Limits can be set here or as Query Result Limits set in Relationship Analysis Client **General Settings**; if the limits are not the same, the lower limit will prevail.

5. Complete the Selection tab.

- Click **All entities**, **All relationships**, or **Specify starting entities** to identify what you want to query against. The Specify starting entities selection allows you to determine at what point in the model you want to begin your search. For instance, if you are looking at a model that depicts world-wide terrorist activity during specific years, you might have country names for entities. Rather than query against the entire model, you might want to look at activity just in Afghanistan. In this case you could select "All" as the **Search type** and "Country" as the **Property name**, leave **Literal** selected, and enter "Afghanistan" as the **Property value**.

Note: The value in the **Property value** field is case sensitive if that field was indexed with the Exact type selected (versus Case insensitive). For more information on selecting types when indexing fields, see **The Entities Tab** on page 211.

You could also click **Field** and select "Location" as the Property value, for example, rather than entering a specific value. If you select Field, an **Input Data** grid containing the name of the field you just selected will appear under the query name along with a cell where you can enter the default value. If you reuse this query elsewhere, you can use the default value you provided in this step, or you can override the default at that time.

- If you clicked **All entities** or **Specify starting entities**, select the **Entity types** for your query. You can choose to query selected types or all types. Click **Select None** to deselect any selected types. In addition to returning a more focused set of results, selecting entity types will affect other factors such as which properties and fields are available in the first step of the query, which directions, entity types, and relationship labels are available in subsequent steps of the query, and so on.
- If you clicked **Specify starting entities**, select the **Search type**:

Exact	Searches the index for data that matches exactly what you enter on the Selection tab, including casing. As with property values, the value here is case sensitive if that field was indexed with the Exact type selected; if case sensitivity was used and you search for "texas" while your data includes entries of "Texas", they will not be returned.
-------	---

Starts with	Searches the index for data that contains text beginning with what you enter on the Selection tab. The search does not need to be a complete word. For example, a literal property value of "tech" or "tec" would be considered a match for a property value containing "Technical", "Technology", "Technologies", or "Technician".
-------------	---

Ends with	Searches the index for data that contains text ending with what you enter on the Selection tab. The search does not need to be a complete word. For example, a literal property value of "Emirates" or "tes" would be considered a match for a property value containing "United Arab Emirates".
Contains	Searches the index for data that contains the text that you enter on the Selection tab. The search string can incorporate any portion of an expected match. For example, a literal property value of "Light" and "Light Amplification" would be considered a match for a property value containing "Light Amplification by Stimulated Emission of Radiation".
Any	Searches the index for data that contains any of the text that you enter on the Selection tab. For example, a literal property value of "Austin Tex" would be considered a match for a property value containing "Texarkana" or "Stephen F. Austin University".
All	Searches the index for data that contains all of the text that you enter on the Selection tab. For example, a literal property value of "Allstate claim 2013" would be considered a match for a property value containing "filed claim with Allstate June 2013", as would literal property values of "all state" or "all 13".
Between	<p>Searches the index for data that falls within a range that you specify on the Selection tab. When you use this search type, you must select a Property name that contains date, time, date/time, or numeric data. All numeric data types are supported with the exception of BigDecimal. For example, the following specifications would return all entities with StartDate values occurring in the year 2000:</p> <ul style="list-style-type: none">• A Property name of "StartDate" that is a Date type• A literal Start value of "1/1/2000"• A literal End value of "12/31/2000"

Fuzzy

Searches the index for the text you enter on the Selection tab but allows for some differentiation (missing letters, extra letters, or substitutions of letters). The amount of differentiation that is acceptable to still be considered a match depends on what you enter in the **Metric** field. This figure must be greater than zero and less than one; in other words, it must range from ".1" to ".9". For example, if you search for "Barton" and enter ".9" as the metric, the search will return records with "Carton" (replaces B with C), "Bartons" (adds s), and "Baton" (removes r), because all of these words are one character different from the search word "Barton".

Wildcard

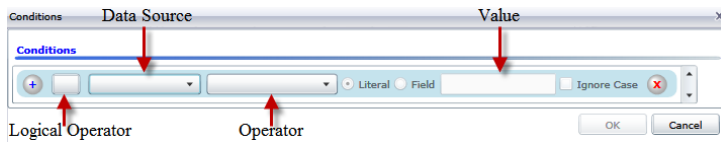
Searches the index for the text you enter on the Selection tab but allows for a single wildcard character or a wildcard character sequence. Supported wildcard characters include the question mark (?), which matches any single character, and the asterisk (*), which matches any character sequence (including blanks). For example, if you search cities in Texas for "Aus*", the search will return records with "Austin", "Austonio" and "Austwell". If you conduct a similar search for "Aust??", only "Austin" will be returned because each question mark represents a single character and the other two city names have more characters in their name.

Note: A query that includes an asterisk wildcard as the first character in the search string may result in a lengthy response time.

- If you clicked **Specify starting entities**, select the **Property name** from the drop-down list. This list contains all properties associated with the entities and relationships that make up the model.

Note: You can only query properties that have been indexed; non-indexed properties will not appear in the Property name drop-down.

- If you clicked **Specify starting entities**, and selected a search type other than "Between", select the **Property value**. You can click **Literal** and enter a text string to be used in the search. Alternatively, you can click **Field** and select the field whose data should be searched; if you choose this option you will also need to enter a value in the Input Data grid.
 - If you clicked **Specify starting entities**, and selected the "Between" search type, select the **Start value** and **End value** to enter the range. You can click **Literal** and enter a value to be used in the search. Alternatively, you can click **Field** and select the field whose data should be searched; if you choose this option you will also need to enter a value in the Input Data grid.
6. Complete the **Conditions** tab if you want to place additional constraints on the query. The Conditions tab has four entry fields:



- If you are creating the first condition, the **Logical operator** field will remain empty. If you are creating a subsequent condition, specify whether this condition should be used in conjunction with previous conditions ("And") or if it should be used instead of previous conditions ("Or").
- Select the property on which the condition will be based in **Data source** field.
- Select an operator for the condition that is appropriate for the data type in the **Operator** field:

Equals Searches model elements for properties with values that match exactly what you enter in the Value field. This can be a numeric value or a text value.

Not Equals Searches model elements for properties with values that have any value other than what you enter in the Value field. This can be a numeric value or a text value.

Exists Searches the model elements for the existence of the property that you select in the Data Source field.

Does not Exist Searches the model for elements that do not contain properties that you select in the Data Source field

Is Blank Searches model elements for properties that contain no data. If a property value is blank, that element will be returned. This can be a numeric value or a text value.

Is Not Blank Searches model elements for properties that contain any data. If a property value is not blank, that element will be returned. This can be a numeric value or a text value.

Greater Than Searches model elements for properties whose values are greater than the value you specify. This can be a numeric, date, date/time, or time value.

Greater Than or Equals Searches model elements for properties with numeric values that are greater than or equal to the value you specify. This can be a numeric, date, date/time, or time value.

Less Than Searches model elements for properties with numeric values that are less than the value you specify. This can be a numeric, date, date/time, or time value.

Less Than or Equals Searches model elements for properties with numeric values that are less than or equal to the value you specify. This can be a numeric, date, date/time, or time value.

Contains Searches model elements for properties with values that contain what you enter in the Data Source field. The search does not need to be a

	complete word. For example, a literal property value of “Light” or “Light Amplification” would be considered a match for a property value containing “Light Amplification by Stimulated Emission of Radiation”, so it would be returned. This can be a numeric value or a text value.
Does not Contain	Searches model elements for properties that do not contain what you enter in the Data Source field. The search does not need to be a complete word. For example, a literal property value of “Light” or “Light Amplification” would be considered a match for a property value containing “Light Amplification by Stimulated Emission of Radiation”, so it would not be returned. This can be a numeric value or a text value.
Starts With	Searches model elements for properties whose values start with the text you enter in the Data Source field. For example, if you enter "Van" for the LastName field it would return results with "Van Buren", "Vandenburg", or "Van Dyck".
Does Not Start With	Searches model elements for properties whose values do not start with the text you enter in the Data Source field. For example, if you enter "Van" for the LastName field it would not return results with "Van Buren", "Vandenburg", or "Van Dyck" but would return results with "Eddie Van Halen".
Ends With	Searches model elements for properties whose values end with the text you enter in the Data Source field. For example, if you filter for records that end with "burg" in the City field, it would return results with "Gettysburg", "Fredricksburg", and "Blacksburg".
Does Not End With	Searches model elements for properties whose values do not end with the text you enter in the Data Source field. For example, if you filter for records that end with "burg" in the City field it would not return results with "Gettysburg", "Fredricksburg", and "Blacksburg" but would return results with "Burgess".
Match Regular Expression	Searches the model elements for properties having a regular expression match for what you enter in the Data Source field. Regular expression matches identify strings of text of interest, such as particular characters, words, or patterns of characters. The value field should contain a valid regular expression pattern.
Is Roughly Similar To	Searches model elements for properties with values close to what you enter in the Data Source field but allows for some differentiation (missing letters, extra letters, or substitutions of letters). This operator is equivalent to the Fuzzy search type with search metric of .5.
Is Similar To	Searches model elements for properties with values close to what you enter in the Data Source field but allows for some differentiation (missing letters, extra letters, or substitutions of letters). This operator is equivalent to the Fuzzy search type with search metric of .6.

Is Very Similar To Searches model elements for properties with values close to what you enter in the Data Source field but allows for some differentiation (missing letters, extra letters, or substitutions of letters). This operator is equivalent to the Fuzzy search type with Search Metric of .7.

d. In the drop-down box following the list of operators:

- Select **Literal** and enter a text string the fourth box (called the **Value** field) to be used in the query.
- Select **Field** and then select the field whose data should be searched in the **Value** field.
- Select a previous step (such as "**Root**" or "**Step1**") and then a property in the **Value** field to compare property values for the current step against values returned in a previous step. (Note that if you named the output on the Output tab of previous steps, those names will appear in the drop-down rather than "Root" or "Step1".) In this case, the properties shown in the Value field are based on properties for the previous step. For example, if you knew the name of one person (Mohamed Atta) who attended a particular event (a meeting in Kandahar) but wanted to know the names of the other attendees, you could create the following query that includes a property value comparison:
 - A root step that looks for an entity type Person with an `_stp_id` of that contains "Mohamed"
 - An Entity to Relationship step with a relationship label of "Attended"
 - A Relationship to Entity step with a condition that includes an Event that contains "Kandahar"
 - An Entity to Relationship step with a relationship label of "Attended" plus a condition that this step's `_stp_id` does not contain the same `_stp_id` value that was found in the root step.

This query will find that Mohamed Atta attended an event in Kandahar and that it was also attended by three other people whose `_stp_id` value is not "Mohamed".

- e. Click **Ignore Case** if the query results can be either upper or lower cased.
- f. Repeat steps a through e to add additional conditions.
- g. Click **OK**.

If, for example, you want to target terrorist activity in Afghanistan between 2001 and 2010, you would create two conditions. First, you would select "date" for the **Property name**, then "Greater Than or Equals", leave Literal selected, and then enter "2001". You would follow this with a second condition set to "And" that also uses "date," then "Less Than or Equals", then "2010". Alternatively, you could click **Field** and select "Date" rather than entering a specific value. Add, delete, or change the order of conditions by using the icons on either side of the conditions. Click **Ignore Case** if the query results can be either upper or lower cased.

7. Complete the **Output** tab to define how you want your output to appear.

- Click the **Include in results** box if you want the results from this step to be included in the output.

Note: This box must be checked for the last step in any series; therefore, if there is only one step you cannot uncheck this box.

- Click **Specify name** and enter text in the **Name** field to provide a name for this step in the output. Click **List** to use this entry as the name and type of the field in hierarchical output; leave it unchecked to have this entry added as a prefix for all output fields. Using the example from step 3, you might call this step "Afghanistan". Output fields from this step may be named "Afghanistan.Latitude" or "Afghanistan.Date".
 - Click **Use type name** to use the field type as the name for this step in the output. Entities will use entity types and relationships will use relationship labels. Continuing with the same example, output fields with this selection may be named "Person.Latitude" or "Person.Date". If you select this option and enter a name in the **Name** field, that name will also be added as a prefix for all output fields in addition to the field type. Continuing with the same example, output fields with this selection may be named "Afghanistan.Person.Latitude" or "Afghanistan.Person.Date".
8. Specify the steps you want the query to take by selecting the appropriate option in the **Add Operations** drop-down. You can complete this step for the Flow, Conditions, or Output tab. Note that your options vary by whether the root element is an entity or a relationship.
- If you choose **Entity to Entity** (valid for All entities and Specify starting entities), you can then refine your search to return data based on relationship labels between two entities (Connected), before entities (Predecessors), or after entities (Successors). For example, if you are querying a model of family members, and you choose a Relationship label of "Father," a Connected query will return all entities that have a Father label between them (in other words, fathers, sons, and daughters). A Predecessors query will return all entities who are a source entity of a Father relationship connected to another entity (in other words, fathers). A Successors query will return all entities who are the target entity of a Father relationship connected to another entity (in other words, sons and daughters).

As in the root step of your query, you can also select Entity types for this step of the query. You can choose to query selected types or all types. Click Select None to deselect any selected types.

- If you choose **Entity to Relationship** (valid for All entities and Specify starting entities), your options are very similar to those for Entity to Entity. You can refine your search to return data based on relationship labels that attach two entities (Connected), occur before entities (Predecessors), or occur after entities (Successors). You can also add conditions to and define output for the query.
- If you choose **Relationship to Entity** (valid for All relationships), you can refine your search to return data based on conditions you set. You can return data when a condition is in place for an entity that is connected to another entity (Connected), an entity that is a source to a relationship (Predecessors), and for when an entity is a target of a relationship (Successors). As in the root step of your query, you can also select Entity types for this step of the query. You can choose to query selected types or all types. Click Select None to deselect any selected types.

Regardless of the type of operation you add, you can create Conditions for that operation. You can also define how you want the output from this step to appear. You will notice that steps subsequent to the root step are given a path for output. The path and the step name define the hierarchy of your output data. If you checked the List box in the root step, this path will default

to being part of the path in the step before it; however, you can remove the name of the root step. For example, if you named the root step "Locations" and clicked the List box, the first step would by default show "/Locations" in the **Path** field. (Alternatively, you could remove "Locations" and leave just the slash to have this step originate at the root.) If you called the first step "CountryName", the second step would by default show "/Locations/CountryName" in the **Path** field and the Locations field output would contain a list of CountryName results. Click the **Include in results** box if you want the results from this step to be included in the output. Click **Dedup** if you want the query to remove duplicate results from the output.

9. Click **OK**.

The Fields Tab

The **Fields** tab allows you to designate fields to be returned and put into the output of your dataflow. If you are building a custom script, you may add, modify, or remove fields. If you are using the query builder the fields are populated for you, but you can change the name and properties of the entities and relationships that were auto-populated.

1. Create your query on the **Query** tab.

Note: This step can be completed before or after the **Fields** tab is complete.

2. Click the **Fields** tab.
3. Click **Add** to open the **Add Input Field** dialog box.
4. Use the **Fields** drop-down and the **Add** button to select the fields you want to query in the model. The fields available for selection depend on the fields assigned in the dataflow input stage. Spectrum supports both simple and complex data types; you can use fields deep within the hierarchical structure of your input file in your query.
5. Click **Close** when you're done adding input fields.
6. Click **Add** to open the **Add Output Field** dialog box.
7. Select the type of output field you want to add from the **Type** drop-down box. The following data types are supported:

Data Type	Description
boolean	A logical type with two values: true and false.
double	A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is 4.9E-324 to 1.7976931348623157E308. For information on E notation, see: http://en.wikipedia.org/wiki/Scientific_notation#E_notation

Data Type	Description
relationships	<p>The links between entities; the factor they share with other entities. If you choose this type of output field, you can change field names and determine which relationships to include upon output. You will need to include in your Gremlin script a command to retrieve a list of relationships and assign them to that data type, as shown in this example:</p> <pre>data["Variants"]=g.idx('entities')[['Name':name]].bothE</pre> <p>Note: If you create an output field this way, Query Model will auto-populate the schema for you.</p>
float	<p>A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{23}) \times 2^{127}$. In E notation, the range of values is 1.4E-45 to 3.4028235E38. For information on E notation, see:</p> <p>http://en.wikipedia.org/wiki/Scientific_notation#E_notation</p>
integer	<p>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).</p>
long	<p>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)</p>
entities	<p>The individual entities that have the relationships you are identifying. If you choose this type of output field, you can change field names and determine which entities to include upon output. You will need to include in your Gremlin script a command to retrieve a list of entities (or vertices) and assign them to that data type, as shown in this example:</p> <pre>data["Variants"]=g.idx('entities')[['Name':name]].both</pre> <p>Note: If you create an output field this way, Query Model will auto-populate the schema for you.</p>
string	<p>A sequence of characters.</p>

You can also add a new, user-defined, custom data type if necessary. Among other types of data, a new type can be a list of any defined data type (though for string, double, integer, long, float, or Boolean types you would select that type from the drop-down options and click the List check box). You can also select an output field based on entities and relationships in your model. If you create an output field this way, Query Model will auto-populate the schema for you. Additionally, you would need to retrieve the entities and/or relationships from Gremlin and then iterate over them, creating a loop. This would result in you building up the fields' child rows and assigning data as you go.

8. Specify a name for the output field you want to add in the **Name** field.
9. Click the **List** check box if you created a custom data type in the form of a list.
10. When you are done adding output fields, click **Close**.
11. Click **OK**.

Input/Output Requirements

The Query Model stage requires that the input stage of your dataflow has defined the input fields that are accessed using data ("input"). Furthermore, any input fields or output fields accessed using the data command need to be defined in the **Fields** tab in the input and output stages. Otherwise, they will not appear as input and output fields in other stages in your dataflow.

Split Entity

The Split Entity stage splits one entity into two or more new entities.

A dataflow that uses a Split Entity stage requires an input that identifies the entity you are splitting. It also has two optional output ports: one contains data for the split entities and the other contains data for the records that were not split.

To configure the Split Entity stage, you need to select the Context Graph model you want to modify and then complete the **Options** tab and possibly the **Runtime** tab, depending on which write mode you want to use.

Input

The Split Entity stage requires that your dataflow include an input stage that contains data from or queries the same model whose element you are splitting. This requirement could be met by a Read from Model stage, a Query Model stage, a source stage of some kind, or even a control stage that combines multiple other input stages.

Options

The Options Tab

The **Options** tab enables you to configure the element that will be split in your model.

1. Select the model whose entities you want to split in the **Model** drop-down box.
2. Select the field that contains the entities you want to split in the Source **ID** field. This is often, but not always, the `_stp_id` field if the ID and type are together, or the `_stp_label` field if the ID and type are separate.
3. Select the type of entity you want to split in the Source **Type** drop-down.
4. Select or enter the type for the newly created entities in the Target **Type** drop-down.
5. Check **Replace label with data from input field** and select the appropriate field if you want to change the label of the newly created entities.
6. Select the relationships you want to be included with the split entities.
7. Repeat steps 4 through 6 if you want to create additional entities from the source entities.
8. Click **OK**.

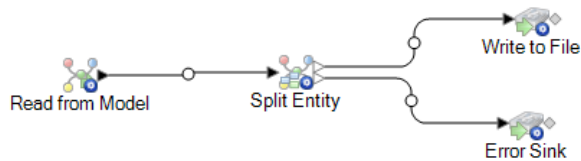
The Runtime Tab

The **Runtime** tab allows you to control processing options.

1. Click **Concurrent writes** if you want to allow the model to be written to by multiple Context Graph stages at the same time. Click **Exclusive lock** (default) if you do not want to allow the model to be written to by multiple Context Graph stages. When this mode is checked, properties can be updated after they are created.
2. Check the **Remove orphans** box to have entities with no relationships after the split removed from the model.
3. Click **OK**.

Output

The Split Entity stage has two optional outgoing ports to which you can attach various sink stages. One sink captures data for successfully split entities and their properties, while the other is used to collect data for the records that were not split. This is called the Error Port, and records that pass through this port into the sink are considered malformed.



Capturing malformed records can help you identify the problem with those records. When you attach a sink to the Error Port, the resulting output file will contain a superset of the fields from both input files. It will also contain a Reason field that specifies why the record failed. So, for example, if your entities input file contains Type, ID, and Location fields, and your relationships input file contains Type, ID, and Label fields, your output file would contain Reason, Type, ID, Location, and Label fields.

Causes for record failure include, but are not limited to, the following:

- The ID field value is empty.
- The ID entered does not return any entities from the selected model.
- The ID and Type combination entered does not return any entities from the selected model.
- The ID and Type combination does not use a valid format (the required format is TYPE_VALUE:ID_VALUE).

Write to Model

The Write to Model stage uses input data to create or update a Context Graph model that contains entities, relationships, and properties.

The Write to Model stage can be used to link data together, resulting in a complex network of relationships called a graph, which can be displayed in a Context Graph model. Once that model is created, it can be queried against in a Read From Model stage or a Query Model stage, or it can be

visualized in Context Graph Visualization workspaces or in the Relationship Analysis Client to identify relationships and trends that may otherwise be hard to find.

To configure a Write to Model stage, you need to complete the **Entities** tab and the **Relationships** tab. You can also use the **Options** tab to set processing preferences and determine how you want data to be written to the graph database.

For more information, see [Sample Model to Context Graph Dataflow](#) on page 216 for examples of configuring Write to Model in a dataflow from start to finish using a flat file and an XML file.

Input

The Write to Model stage requires that your dataflow contain an input stage with defined fields that you can use to create a Context Graph model.

The Entities Tab

The Entities tab allows you to configure how entities are created and updated. These entities can represent objects or events and will be stored in your model. You can use the Relationships tab to link them to other entities and create relationships. To create entities for your model, you will need to complete the following information:

1. Enter the name of your model in the **Model** field.
2. Click **Add...** to create a new entity. The **Add Entity** dialog box will appear.
3. Select the field name to be used to generate the Entity ID in the **Input field** field.
4. If you want the entity to have a name other than what is automatically provided, change the contents of the **Type** field to the desired name.
5. On the **Properties** tab, you define which properties you want to be included with the entity type you are creating.

For example, if you are creating an entity type that represents places, you might want to choose latitude and longitude as properties. You can select **Input** or **Metadata**, depending on what you want to use to define the properties. If your model does not contain metadata, that option will be disabled.

Option	Description
Input	<p>The grid is populated with input fields, which are shown in the Field column; there is one row for each input field. Select the fields you want to be included as properties for the entity type by checking the Include box for that field. The Name column represents the name you want to property to have in the model; it defaults to the input field name, but you can select from any property in the Name drop-down or manually enter a name.</p> <p>Note: Spectrum supports both simple and complex data types; you can use fields deep within the hierarchical structure of your input file as an entity. If you are using hierarchical data, you will also see a Filter control that allows you to filter out data on the Property list based on the path of the field. Likewise, you will see a control that allows you to hide non-primitive fields.</p>

Option	Description
Metadata	The grid is populated with properties from the metadata, which are shown in the Name column; there is one row for each property. The properties shown are determined by whether you are defining a known entity type. If it is a known type, the Name column will include properties specific to that type; if it is not a known type, the Name column will include a list of all properties in the model. Select the properties you want to populate for the entity type by checking the Include box for that property. The Field column contains names of input fields you can map properties to and whose data can be used to populate properties. If an input field matches the property name, it will automatically be mapped and the Include box will be checked.

6. Click the **Updates** tab and select how you want Write to Model to manage updates. After an entity is created it can be updated over time when data with the same ID is input into the Write to Model stage.

Option	Description
Always update properties	Properties are always updated with the most recent information. This includes updating with null or empty strings.
Update properties unless all input is null	Properties are always updated unless all input fields associated with the selected properties are null.
Never overwrite properties with empty data	Properties are always updated unless the input is a null or empty string.
Never overwrite non-empty properties	Properties are never updated once populated with non-empty data.

7. Repeat steps 4 on page 211 through 6 on page 212 to add additional entities. When you are done adding entities, click **Close**.
8. Select which fields you want to be indexed in your model by clicking the **Indexes...** button and checking the box for those fields.

In the **Type** column, you can choose whether the data should be indexed exactly as-is, with case sensitivity, or if it should be indexed without case sensitivity, which typically results in greater response to a search.

Note: The `_stp_id`, `_stp_type`, and `_stp_label` properties are internal properties and will always appear in the list of indexed fields. You can deselect `_stp_type` and `_stp_label`, but `_stp_id` must be indexed; however, you are able to designate whether its index type should be exact or with case insensitivity.

Selecting which fields to index, rather than indexing all fields in your model, results in faster performance when writing to a model. However, if you later attempt to query fields in your model that were not indexed, the response time will be slower.

For example, the **Specify starting entity** option in the query tool for the Relationship Analysis Client works only on indexed properties. You can query non-indexed properties using conditions, but the performance will be slower.

The Relationships Tab

Provide information described in this procedure to create relationships for your model.

The **Relationships** tab allows you to configure how relationships are created between source and target entities. These relationships represent the connection between two entities (for example, John knows Mary).

1. Click **Add**
2. Select the entity to act as a source for your relationship in the **Source** field.
3. Click the appropriate label type for your relationship: **String** or **Field**. If string, enter the string in the text box underneath. If field, select the field you want to use to generate the label for this relationship in the drop-down box.
4. Optional: To allow a relationship to be created more than once between a source and target entity, click the **Allow more than one relationship based on unique ID** box and select the field on which to base the relationship in the drop-down box.
5. Select the entity to act as a target for your relationship in the **Target** field.
6. On the **Properties** tab, define which properties you want to be included with the relationship you are creating.

For example, if you are creating a relationship called "treated" between an entity type of "doctor" and an entity type of "patient," you might want to choose date and diagnosis as properties. You can select **Input** or **Metadata**, depending on what you want to use to define the properties. If your model does not contain metadata, that option will be disabled.

Option	Description
Input	The grid is populated with input fields, which are shown in the Field column; there is one row for each input field. Select the fields you want to be included as properties for the relationship by checking the Include box for that field. The Name column represents the name you want to property to have in the model; it defaults to the input field name, but you can select from any property in the Name drop-down or manually enter a name.
Metadata	The grid is populated with properties from the metadata, which are shown in the Name column; there is one row for each property. The properties shown are determined by whether you are defining a known relationship. If it is a known relationship, the Name column will include properties specific to that relationship; if it is not known, the Name column will include a list of all properties in the model.

Option	Description
	Select the properties you want to populate for the relationship by checking the Include box for that property. The Field column contains names of input fields you can map properties to and whose data can be used to populate properties. If an input field matches the property name, it will automatically be mapped and the Include box will be checked.

7. Optional: Click **Add** to add a new condition.

On the **Conditions** tab, specify conditions that control when a relationship is created between a source and target entity.

- a) If you are creating the first condition, the **Logical operator** field will be grayed out. If you are creating a subsequent condition, specify whether this condition should be used in conjunction with previous conditions or if it should be used instead of previous conditions.
- b) Select the element on which the condition will be based in the **Data source** field.
- c) Select the field that the condition will be based in the **Field name** field.
- d) Select the operator for the condition in the **Operator** field.
- e) Enter the value for the condition in the **Value** field and click **Add**.
- f) Repeat steps **7.a** on page 214 through **7.e** on page 214 to add additional conditions.
- g) When you are done adding conditions, click **Close**.

Completing this step displays the **Add Condition** dialog.

8. Click the **Updates** tab.

9. Select the appropriate action for updating and overwriting properties.

After an relationship is created it can be updated over time when data with the same source and target ID is input into the Write to Model stage. This selection determines how Write to Model manages updates.

Option	Description
Always update properties	Properties are always updated with the most recent information. This includes updating with null or empty strings.
Update properties unless all input is null	Properties are always updated unless all input fields associated with the selected properties are null.
Never overwrite properties with empty data	Properties are always updated unless the input is a null or empty string.
Never overwrite non-empty properties	Properties are never updated once populated with non-empty data.

10. Click **Add**.
11. Repeat steps **1** on page 213 through **10** on page 215 to add additional relationships.
12. When you are done adding relationships, click **Close**.

The Options Tab

Write mode

- Click **Initial load** if you are loading the model for the first time. The model will be locked and unable to be written to by other Write to Model stages. When this mode is checked, the only available option on the Updates tab in the Add Entity dialog box will be **Never overwrite non-empty properties**. Therefore, if you have multiple input files in your dataflow, they can all create properties, but none of them can update existing properties with new values. This mode provides better performance when initially loading a model. Existing data, if there is any, will be cleared prior to writing.
- Click **Concurrent writes** if you want to allow the model to be written to by multiple Write to Model stages at the same time. When this mode is checked, the **Clear model before processing** option is disabled and the model is created prior to running the job. If the model does not already exist, it will be created when the stage is closed.
- Click **Exclusive lock** (default) if you do not want to allow the model to be written to by multiple Write to Model stages. When this mode is checked, properties can be updated after they are created.

Clear model before processing

Check this check box if you wish to remove all existing entities and relationships before processing new data. If this is not selected, new information will be used to update any existing entities and relationships.

Note: Using this option does not alter security settings for Context Graph. The model will be recreated, but the security settings will remain the same.

Remove orphaned entities after processing

Check this check box if you wish to remove entities that have no relationships.

Setting Exclusive Lock Timeout Duration

If you select the **Exclusive lock** write mode on the Write to Model **Options** tab, or if you apply a centrality algorithm (degree, betweenness, closeness, or influence) to a model, that model will be locked while those processes are running and any action you attempt on that model that requires write access will result in a timeout until those processes are complete.

You can specify how long a process should wait before timing out by modifying the neo4j properties file. The default is 10 seconds, or 10,000 milliseconds.

1. Edit the `SpectrumFolder\server\modules\hub\db\neo4j.properties` file.
2. Navigate to the `ha.wait.for.exclusive.lock.timeout` entry.

3. Enter the duration, in milliseconds, that you want to wait for the lock to be released or the process to time out.

A value of 0 milliseconds will cause an immediate timeout. Leaving this property blank will cause the server to wait indefinitely.

Note: If a model is currently in use with the **Concurrent writes** write mode selected, and a subsequent process is attempted with the **Exclusive lock** write mode selected, the model will be locked and the latter process will time out according to the settings made here.

Output

The Write to Model stage has one optional outgoing port that collects any records that the dataflow did not process correctly. This is called the Error Port, and records that pass through this port into the sink are considered malformed.

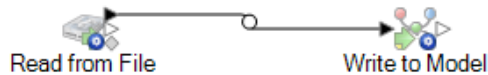
Sample Model to Context Graph Dataflow

This section describes how to configure a simple dataflow that includes a Write to Model stage.

The first example uses a flat file for input, and the second example uses an XML file for input; both files include names of employees and their managers, along with other information described in more detail in the following sections. The end result is the same for both dataflows: a model that depicts the reporting structure of a small organization.

Flat Sample

The Write to Model dataflow that uses a flat file for input looks like this:



Configuring Read from File

The Read from File stage uses a comma-delimited file that includes records with the following fields:

- Employee ID
- Name
- Title
- Manager ID

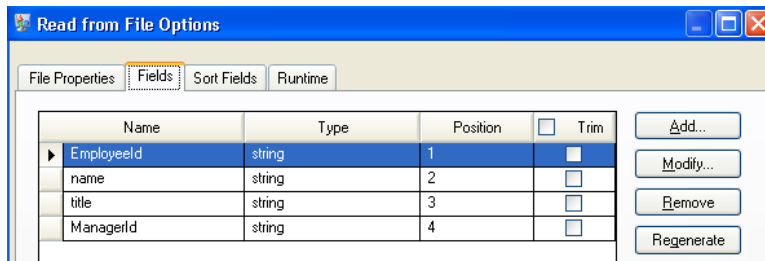
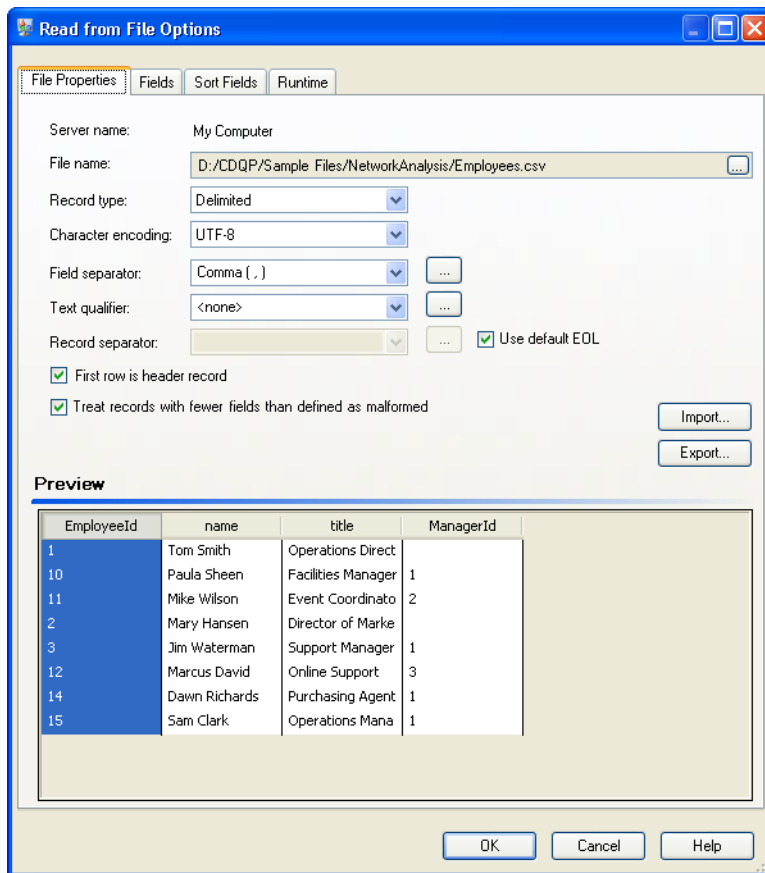
The input file itself looks like this:


```

1 EmployeeId,name,title,ManagerId
2 1, Tom Smith, Operations Director,
3 10, Paula Sheen, Facilities Manager, 1
4 11, Mike Wilson, Event Coordinator, 2
5 2, Mary Hansen, Director of Marketing,
6 3, Jim Waterman, Support Manager, 1
7 12, Marcus David, Online Support, 3
8 14, Dawn Richards, Purchasing Agent, 1
9 15, Sam Clark, Operations Manager, 1
    
```

Notice that two employees do not have manager IDs. These employees (Tom Smith and Mary Hansen) are both directors and therefore have no manager in this exercise. All other employees have a number in the ManagerID field that refers to the employee who is their manager. For example, Paula Sheen's record has "1" in the ManagerID field, indicating that Tom Smith is her manager.

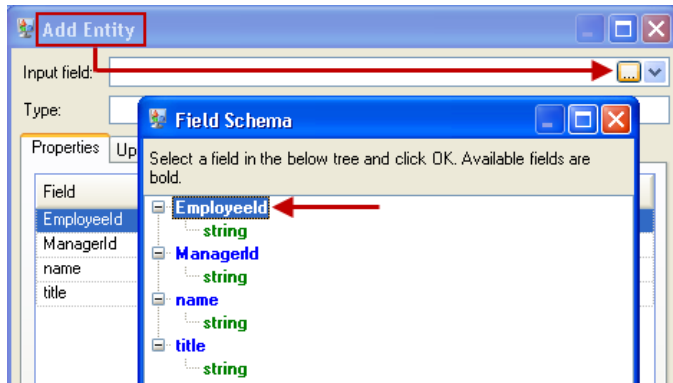
The Read from File stage appears as follows when it is configured to work with this input file:



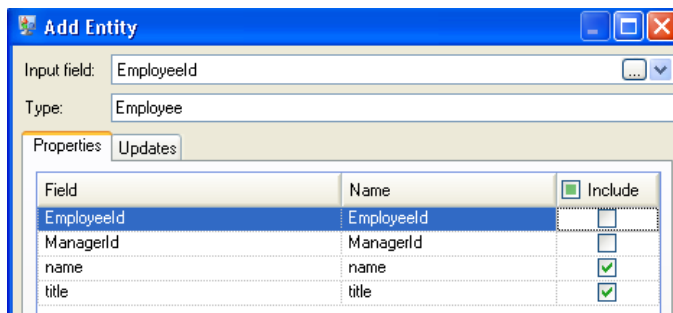
Configuring Write to Model

Next we configure the Write to Model stage. After naming the model "Employees" we configure the stage to include the entities and relationships that will comprise the model.

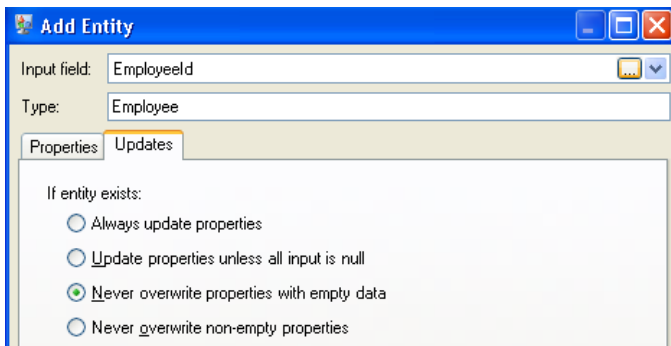
Because we are creating a model that is similar to an organization chart, our entities are employees who are assigned numeric IDs. The first thing we do on the **Add Entity** dialog box is click the browse button to access the **Field Schema** dialog box, and then select "EmployeeId" from the list of available fields. This is the first group of entities in our model.



Next, we set the **Type** field to "Employee" and check the boxes for "name" and "title" because we want the information from those fields to be brought in as properties for the EmployeeID entities in the model.

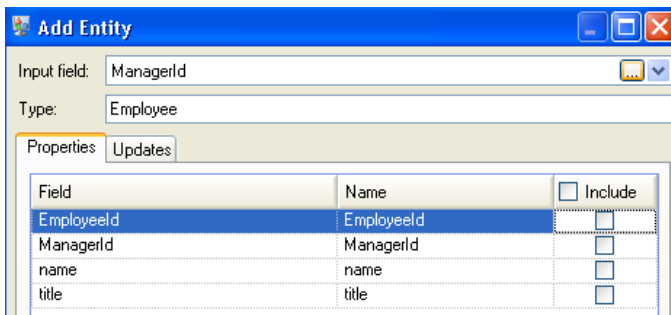


After setting properties for the Employee entity, we configure the processing options. The Updates tab enables you to specify whether properties can be updated in the model once they are in place and if they should overwrite existing data. For instance, in our example, Mary Hansen would be encountered twice because on record 4, she is referred to as an employee, but on record 3, she is referred to as a manager. When Write to Model processes Mary for the second time, it could potentially overwrite or remove data that was populated as a result of the first time it processed Mary. By selecting **Never overwrite properties with empty data** (which is the default), any updates that occur will create new properties and overwrite existing properties, but they will not blank out properties that were set by the first encounter but missing in the second encounter. This also ensures that the order in which these records are read has no impact on the model.

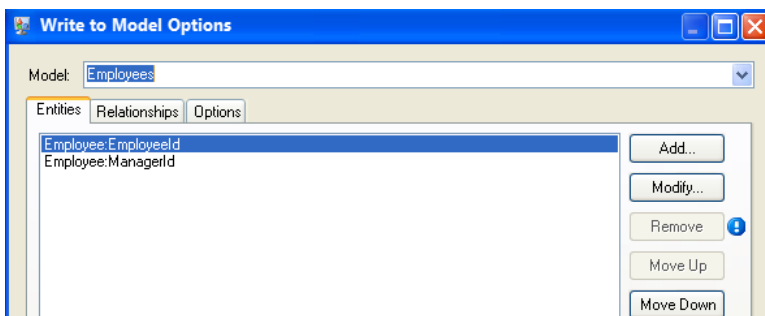


If we selected **Always update properties**, data would always be overwritten and only the last set of property data would be reflected in the model. If we selected **Update properties unless all input is null**, data would always be overwritten unless every field in the new record were blank. Finally, if we selected **Never overwrite non-empty properties**, the first set of data for any given field would be retained, unless that field were blank. In that case, the first set of non-blank data would be retained.

We repeat these steps to add "ManagerId" as the second group of entities in our model. Although ManagerID and EmployeeID are different fields in the input file, both entities' types are set to "Employee." If we set ManagerID to a different type, the model would contain two entities for mid-level managers. For example, Jim Waterman would have an entity as an employee and an entity as a manager. With both entities being set to "Employee" as the type, mid-level managers such as Jim will have just one entity in the model. That entity will have other entities coming into it (from employees) and another entity going out of it (to their respective manager). Note that we do not add properties to the ManagerID entities because the values in those fields (name, title) apply to the employees, not the managers. Also, we accept the **Never overwrite properties with empty data** default selection on the Updates tab.

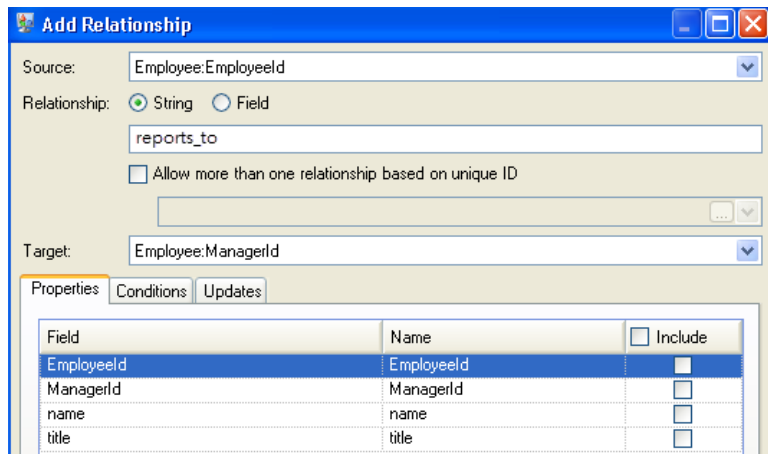


The completed Entities tab for this example appears as follows:

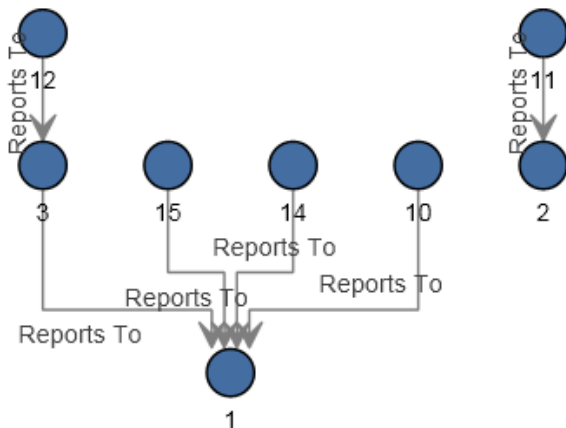


Now we configure the Relationships tab. The first thing we do on the **Add Relationship** dialog box is select the source of the relationship from the list of entities created on the Entities tab. The relationship between our entities reflects the reporting structure (employee to manager); therefore, we select the "Employee:EmployeeID" entity as the source. Next, we select "String" as name of the relationship, and we enter the text "reports_to." After that, we select the target of the relationship from the list of the entities created on the Entities tab; for our example, we select "Employee:ManagerID." If we were using a "manages" relationship instead of a "reports_to" relationship, we would reverse the selections in the source and target fields.

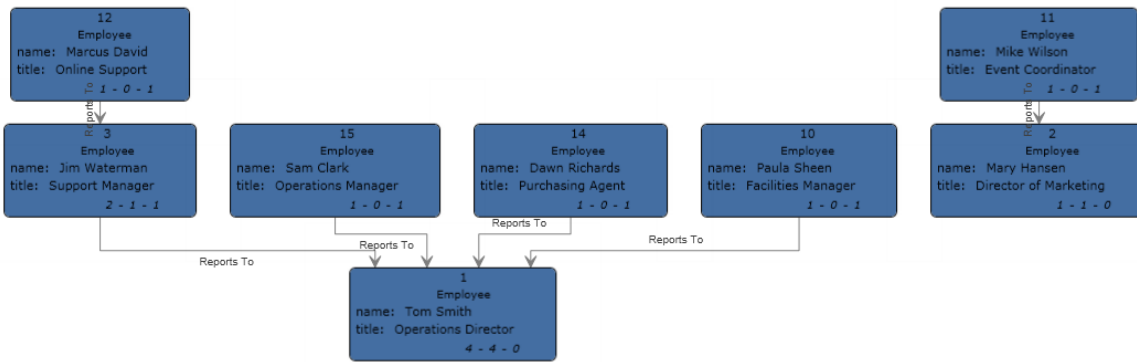
The completed Relationships tab for this example appears as follows:



The configuration of this dataflow is complete and results in the following model, as depicted in the Relationship Analysis Client. This example uses the **Hierarchic** layout with default settings for entities.

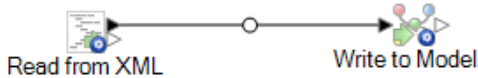


Another way to view this same data is with **Panel** style, as shown below. The benefit of using Panel style is that you can see the properties associated with each entity.



XML Sample

The Write to Model dataflow that uses an XML file for input looks like this:



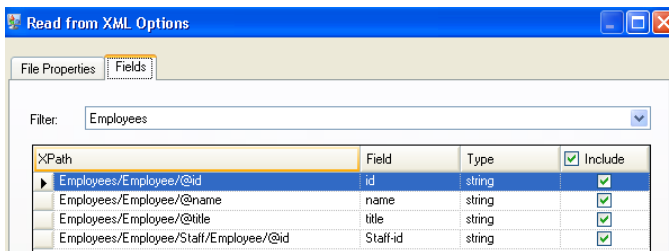
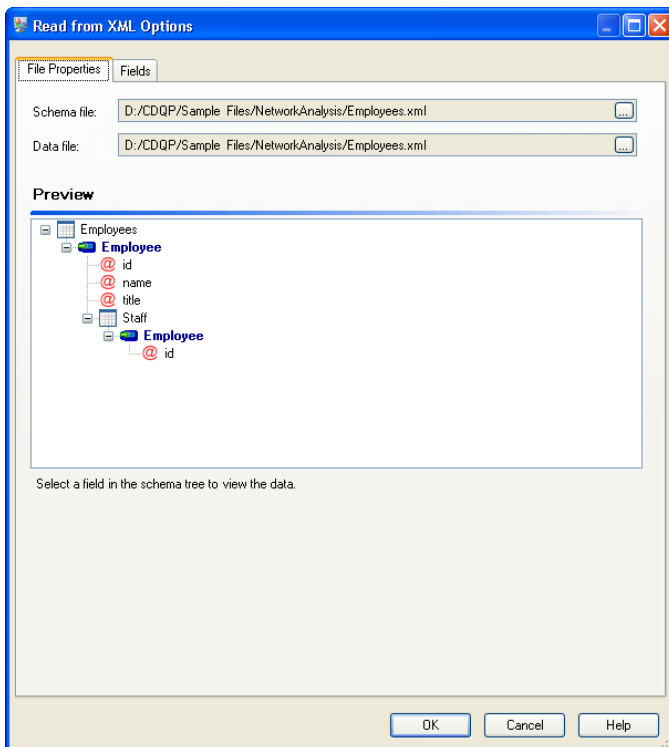
Configuring Read from XML

The Read from XML stage uses a hierarchical file that contains the following:

```

<Employees>
  <Employee id="1" name="Tom Smith" title="Operations Director">
    <Staff>
      <Employee id="3"/>
      <Employee id="10"/>
      <Employee id="14"/>
      <Employee id="15"/>
    </Staff>
  </Employee>
  <Employee id="2" name="Mary Hansen" title="Director of Marketing">
    <Staff>
      <Employee id="11"/>
    </Staff>
  </Employee>
  <Employee id="3" name="Jim Waterman" title="Support Manager">
    <Staff>
      <Employee id="12"/>
    </Staff>
  </Employee>
  <Employee id="10" name="Paula Sheen" title="Facilities Manager"/>
  <Employee id="11" name="Mike Wilson" title="Event Coordinator"/>
  <Employee id="12" name="Marcus David" title="Online Support"/>
  <Employee id="14" name="Dawn Richards" title="Purchasing Agent"/>
  <Employee id="15" name="Sam Clark" title="Operations Manager"/>
</Employees>
    
```

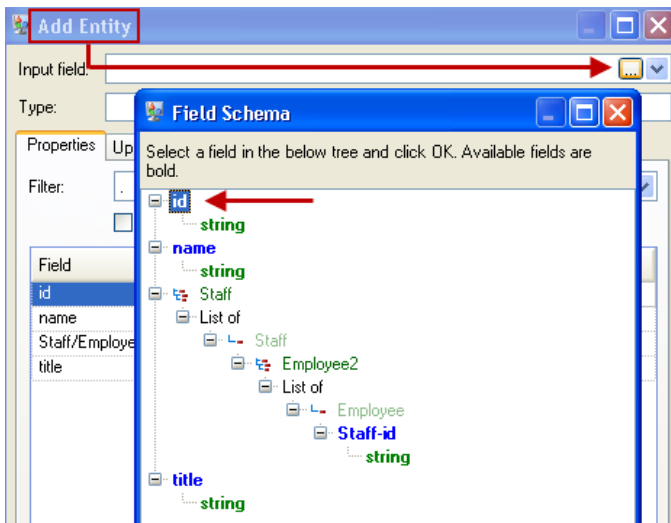
The Read from XML stage appears as follows when it is configured to work with this input file:



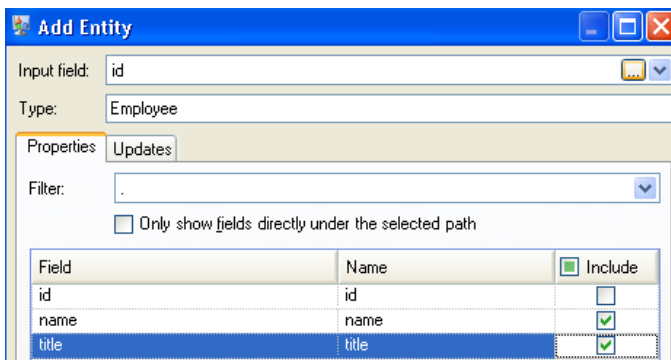
Configuring Write to Model

Next we configure the Write to Model stage. After naming the model "Employees" we configure the stage to include the entities and relationships that will comprise the model.

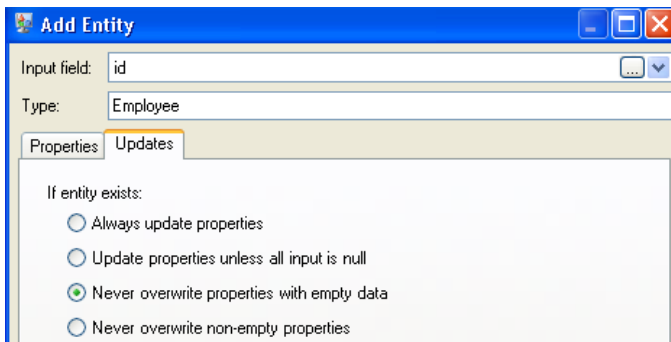
Because we are creating a model that is similar to an organization chart, our entities are employees who are assigned numeric IDs. The first thing we do on the **Add Entity** dialog box is click the browse button to access the **Field Schema** dialog box, and then select "id." This is the first group of entities in our model.



Next, we set the **Type** field to "Employee" and check the boxes for "name" and "title" because we want the information from those fields to be brought in as properties for the ID entities in the model.



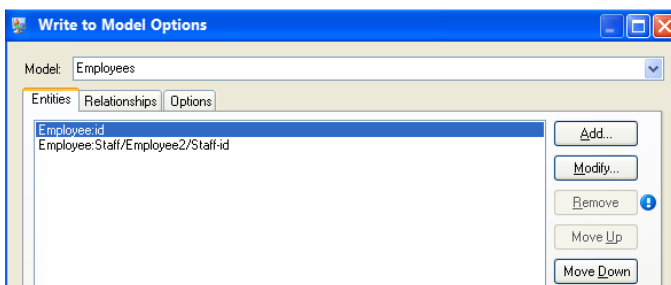
After setting properties for the ID entity, we configure the processing options. The Updates tab enables you to specify whether properties can be updated in the model once they are in place and if they should overwrite existing data. For instance, in our example, Mary Hansen would be encountered twice because for ID 2, she is an employee, but for ID 11, she is a manager. When Write to Model processes Mary for the second time, it could potentially overwrite or remove data that was populated as a result of the first time it processed Mary. By selecting **Never overwrite properties with empty data** (which is the default), any updates that occur will create new properties and overwrite existing properties, but they will not blank out properties that were set by the first encounter but missing in the second encounter. This also ensures that the order in which these records are read has no impact on the model.



If we selected **Always update properties**, data would always be overwritten and only the last set of property data would be reflected in the model. If we selected **Update properties unless all input is null**, data would always be overwritten unless every field in the new record were blank. Finally, if we selected **Never overwrite non-empty properties**, the first set of data for any given field would be retained, unless that field were blank. In that case, the first set of non-blank data would be retained.

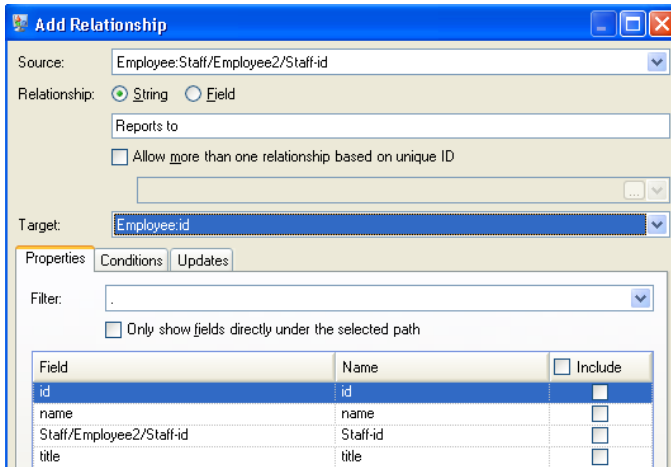
We repeat these steps to add "ManagerID" as the second group of entities in our model. Although ManagerID and EmployeeID are different fields in the input file, both entities' types are set to "Employee." If we set ManagerID to a different type, the model would contain two entities for mid-level managers. For example, Jim Waterman would have an entity as an employee and an entity as a manager. With both entities being set to "Employee" as the type, mid-level managers such as Jim will have just one entity in the model. That entity will have other entities coming into it (from employees) and another entity going out of it (to their respective manager). Note that we do not add properties to the ManagerID entities because the values in those fields (name, title) apply to the employees, not the managers. Also, we accept the **Never overwrite properties with empty data** default selection on the Updates tab.

The completed Entities tab for this example appears as follows:



Now we configure the Relationships tab. The first thing we do on the **Add Relationship** dialog box is select the source of the relationship from the list of entities created on the Entities tab. The relationship between our entities reflects the reporting structure (employee to manager); therefore, we select the "Employee:Staff/Employee/Staff-id" entity as the source. Next, we select "String" as name of the relationship, and we enter the text "Reports to." After that, we select the target of the relationship from the list of the entities created on the Entities tab; for our example, we select "Employee.id." If we were using a "manages" relationship instead of a "reports to" relationship, we would reverse the selections in the source and target fields.

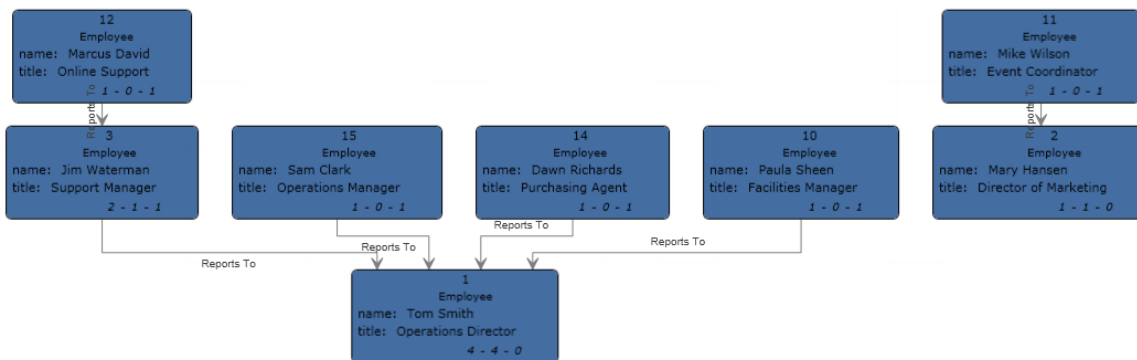
The completed Relationships tab for this example appears as follows:



The configuration of this dataflow is complete and results in the following model, as depicted in the Relationship Analysis Client:



As with the flat sample, this model can also be viewed in **Panel** style, as shown below.



Data Normalization stages

Advanced Transformer

The Advanced Transformer job scans and splits strings of data into multiple fields using tables or regular expressions. It extracts a specific term or a specified number of words to the right or left of a term. Extracted and non-extracted data can be placed into an existing field or a new field.

For example, want to extract the suite information from this address field and place it in a separate field.

2300 BIRCH RD STE 100

To accomplish this, you could create an Advanced Transformer that extracts the term STE and all words to the right of the term STE, leaving the field as:

2300 BIRCH RD

Input

Advanced Transformer uses any defined input field in the data flow.

Options

Advanced Transformer options can be configured at the stage level, through any of the Spectrum Technology Platform clients, or at runtime, using dataflow options.

Configuring Options

To specify the options for Advanced Transformer you create a rule. You can create multiple rules then specify the order in which you want to apply the rules. To create a rule:

1. Double-click the instance of Advanced Transformer on the canvas. The Advanced Transformer Options dialog displays.
2. Select the number of runtime instances and click **OK**. Use the Runtime Instances option to configure a dataflow to run multiple, parallel instances of a stage to potentially increase performance.
3. Click the **Add** button. The Advanced Transformer Rule Options dialog displays.

Note: If you add multiple transformer rules, you can use the **Move Up** and **Move Down** buttons to change the order in which the rules are applied.

4. Select the type of transform action you wish to perform and click **OK**. The options are listed in the table below.

Table 21: Advanced Transformer Options

Option	Description
Source	Specifies the source input field to evaluate for scan and split.
Extract using	<p>Select Table Data or Regular Expressions.</p> <p>Select Table Data if you want to scan and split using the XML tables located at <Drive>:\Program Files\Precisely\Spectrum\server\modules\advancedtransformer\data. See Table Data Options below for more information about each option.</p> <p>Select Regular Expressions if you want to scan and split using regular expressions. Regular expressions provide many additional options for splitting data. You can use the prepackaged regular expressions by selecting one from the list or you can construct your own using RegEx syntax.</p> <p>For example, you could split data when the first numeric value is found, as in "John Smith 123 Main St." where "John Smith" would go in one field and "123 Main St." would go in another. See Regular Expression options below for more information about each option.</p>
Table Data Options	
Non-extracted Data	<p>Specifies the output field that you want to contain the transformed data. If you want to replace the original value specify the same field in the Destination field as you did in the Source drop-down box.</p> <p>You may also type in a new field name in the Destination field. If you type in a new field name, that field name will be available in stages in your dataflow that are downstream of Advanced Transformer.</p>
Extracted Data	<p>Specifies the output field where you want to put the extracted data.</p> <p>You may type in a new field name in the Extracted Data field. If you type in a new field name, that field name will be available in stages in your dataflow that are downstream of Advanced Transformer.</p>

Option	Description
Tokenization Characters	<p>Specifies any special characters that you want to tokenize. Tokenization is the process of separating terms. For example, if you have a field with the data "Smith, John" you would want to tokenize the comma. This would result in terms:</p> <ul style="list-style-type: none"> • Smith • , • John <p>Now that the terms are separated, the data can be split by scanning and extracting on the comma so that "Smith" and "John" are cleanly identified as the data to standardize.</p>
Table	<p>Specifies the table that contains the terms on which to base the splitting of the field. For a list of tables, see Advanced Transformer Tables. For information about creating or modifying tables, see Introduction to Lookup Tables.</p>
Lookup multiple word terms	<p>Select this check box to enable multiple word searches within a given string. For example:</p> <p>Input String = "Cedar Rapids 52401" Business Rule = Identify "Cedar Rapids" in string based on a table that contains the entry; Cedar Rapids = US Output = Identifies presence of "Cedar Rapids" and places the terms into a new field, for example City.</p> <p>For multiple word searches, the search stops at the first occurrence of a match.</p> <p>Note: Selecting this option may adversely affect performance.</p>

Option	Description
Extract	<p>Specifies the type of extraction to perform. Select from one of these:</p> <p>Extract term Extracts the term identified by the selected table.</p> <p>Extract N words to the right of the term Extracts words to the right of the term. You specify the number of words to extract. For example, if you want to extract the two words to the right of the identified term, specify 2.</p> <p>Extract N words to the left of the term Extracts words to the left of the term. You specify the number of words to extract. For example, if you want to extract the two words to the left of the identified term, specify 2.</p> <p>If you choose to extract words to the right or left of the term, you can specify if you want to include the term itself in the destination data or the extracted data. For example, if you have this field:</p> <p>2300 BIRCH RD STE 100</p> <p>and you want to extract "STE 100" and place it in the field specified in extracted data, you would choose to include the term in the extracted data field, thus including the abbreviation "STE" and the word "100".</p> <p>If you select neither Destination nor Extracted data, the term will not be included and is discarded.</p>
Regular Expressions Options	
Regular Expressions	<p>Select a prepackaged regular expressions from the list or construct your own in the text box. Advanced Transformer supports standard RegEx syntax.</p> <p>The Java 2 Platform contains a package called java.util.regex, enabling the use of regular expressions. For more information, go to: java.sun.com/docs/books/tutorial/essential/regex/index.html.</p>
Ellipsis Button	Click this button to add or remove a new regular expression.
Populate Group	After you have selected a predefined or typed a new Regex expression, click Populate Group to extract any Regex groups and place the complete expression, as well as any Regex groups found, into the Groups list.

Option	Description
Groups	<p>This column shows the regular expressions for the selected Regular Expressions group.</p> <p>For example, if you select the Date Regex expression, the following expression displays in the text box: <code>(1[012]{1,2}0?[1-9])[-./]([12][0-9]3[01]{1,2}0?[1-9])[-./]([0-9]{4})</code>. This Regex expression has three parts to it and the whole expression and each of the parts can be sent to a different output field. The entire expression is looked for in the source field and if a match is found in the source field, then the associated parts are moved to the assigned output field. If the source field is "On 12/14/2006" and you apply the Date expression to it, and assign the entire date (such as, "12/14/2006") to be placed in the DATE field, the "12" to be placed in MONTH field, the "14" to be placed in the DAY field and "2006" to be placed in YEAR field. It will look for the date and if it finds it will move the appropriate information to the appropriate output field.</p> <p>Source Field: "On 12/14/2006" DATE: "12/14/2006" MONTH: "12" DAY: "14" YEAR: "2006"</p>
Output Field	Pull-down menu to select an output field.

Configuring Options at Runtime

Advanced Transformer rules can be configured and passed at runtime if they are exposed as dataflow options. This enables you to override the existing configuration with JSON-formatted strings. You can also set stage options when calling the job through a process flow or through the job executor command-line tool.

You can find schemas for AdvancedTransformerRules in the following folder:

```
<Spectrum Location>\server\modules\jsonSchemas\advancedTransformer
```

To define Advanced Transformer rules at runtime:

1. In Enterprise Designer, open a dataflow that uses the Advanced Transformer stage.
2. Save and expose that dataflow.
3. Go to **Edit > Dataflow Options**.
4. In the **Map dataflow options to stages** table, expand Advanced Transformer. Check the box for AdvancedTransformerRules.
5. Optional: Change the name of the options in the **Option label** field.
6. Click **OK** twice.

Output

Advanced Transformer does not create any new output fields. Only the fields you define are written to the output.

Open Parser

Open Parser parses your input data from many cultures of the world using a simple but powerful parsing grammar. Using this grammar, you can define a sequence of expressions that represent domain patterns for parsing your input data. Open Parser also collects statistical data and scores the parsing matches to help you determine the effectiveness of your parsing grammars.

Use Open Parser to:

- Parse input data using domain-specific and culture-specific parsing grammars that you define in Domain Editor.
- Parse input data using domain-independent parsing grammars that you define in Open Parser using the same simple but powerful parsing grammar available in Domain Editor.
- Parse input data using domain-independent parsing grammars at runtime that you define in Dataflow Options.
- Preview parsing grammars to test how sample input data parses before running the job using the target input data file.
- Trace parsing grammar results to view how tokens matched or did not match the expressions you defined and to better understand the matching process.

Input

Open Parser accepts the input fields that you define in your parser grammar. For more information, see [Header Section Commands](#).

If you are performing culture-specific parsing, you can optionally include a CultureCode field in the input data to use a specific culture's parsing grammar for a record. If you omit the CultureCode field, or if it is empty, then each culture listed in the Open Parser stage is applied, in the order specified. The result from the culture with the highest parser score, or the first culture to have a score of 100, is returned. For more information about the CultureCode field, see [Assigning a Parsing Culture to a Record](#).

Options

The following tables list the options for the Open Parser stage.

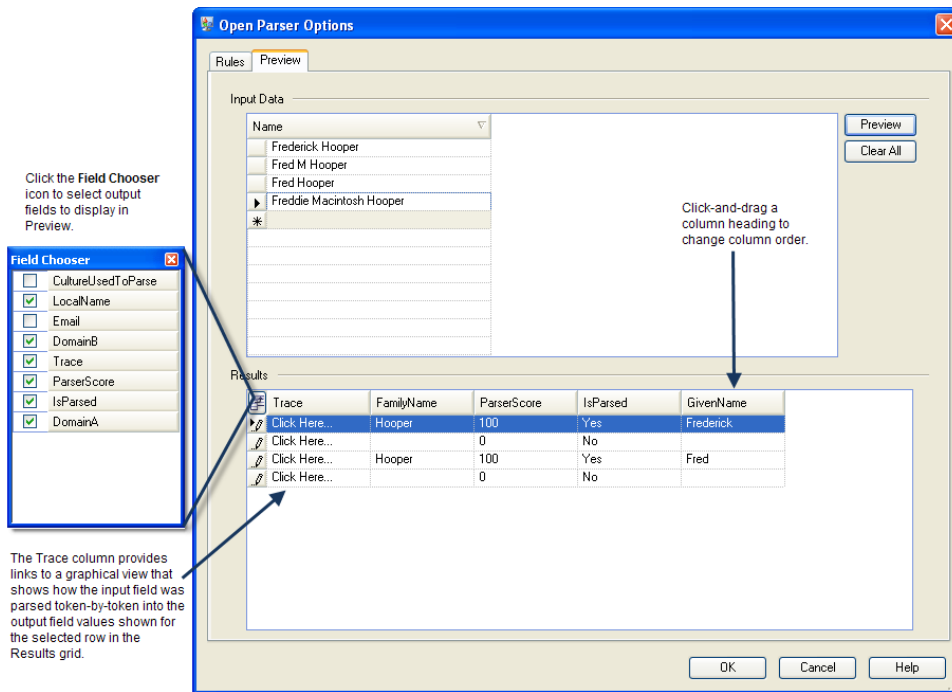
Rules Tab

Option	Description
Use culture-specific domain grammar	<p>Specifies to use a language and domain specific parsing grammar which has already been defined in the Open Parser Domain Editor tool in Enterprise Designer. For more information about defining domains, see Defining a Culture-Specific Parsing Grammar.</p> <p>If you choose this option you will also see these options:</p> <p>Domain Specifies the parsing grammar to use.</p> <p>Cultures Specifies the language or culture of the data you want to parse. Click the Add button to add a culture. You can change the order in which Open Parser attempts to parse the data with each culture by using the Move Up and Move Down buttons. For more information about cultures, see Defining a Culture-Specific Parsing Grammar.</p> <p>Return multiple parsed records Enable this option to have Open Parser return records for each culture that successfully parses the input. If you do not check this box, Open Parser will return the results for the first record that achieves a parser score of 100, regardless of culture. If all cultures run without hitting a record that has parser score of 100, Open Parser will return the record with the score closest to 100. If multiple cultures return records with the same high score under 100, the order set in Step 4 will determine which culture's record is returned.</p>
Define domain-independent grammar	<p>Choose this option if you want to define a parsing grammar that should be applied without consideration of the language or domain of the input data. If you choose this option, the grammar editor will appear and you can define the parsing grammar directly in the Open Parser stage rather than using the Open Parser Domain Editor tool in Enterprise Designer.</p> <p>Note: You can also define domain-independent grammar at runtime. For more information, see Defining Domain-Independent Parsing Grammars at Runtime.</p>

Preview Tab

Creating a working parsing grammar is an iterative process. Preview is useful in testing out variations on your input to make sure that the parsing grammar produces the expected results.

Type test values in the input field and then click **Preview**.



The parsed output fields display in the **Results** grid. For information about the output fields, see [Output](#) on page 233. For information about trace, see [Tracing Final Parsing Results](#). If your results are not what you expected, click the **Rules** tab and continue editing the parsing grammar and testing input data until it produces the expected results.

Output

Table 22: Open Parser Output

Field Name	Description / Valid Values
<Input Field>	The original input field defined in the parsing grammar.
<Output Fields...>	The output fields defined in the parsing grammar.
CultureCode	The culture codes contained in the input data. For a complete list of supported culture codes, see Assigning a Parsing Culture to a Record .
CultureUsedtoParseSelect a match	The culture code value used to parse each output record. This value is based on results in the Match Results List and matches to a culture-specific parsing grammar. then click Remove .

Field Name	Description / Valid Values
IsParsed	Indicates whether an output record was parsed. The possible values are Yes or No.
ParserScoreSelect a match results in the Match Results List and then click Remove .	Indicates the total average score. The value of ParserScore will be between 0 and 100, as defined in the parsing grammar. 0 is returned when no matches are returned. For more information, see Scoring .
Trace	Click this control to see a graphical view of how each token in the parsing grammar was parsed to an output field for the selected row in the Results grid.

Table Lookup

The Table Lookup stage standardizes terms against a previously validated form of that term and applies the standard version. This evaluation is done by searching a table for the term to standardize.

For example:

	First Name	Last Name
Source Input:	Bill	Smith
Standardized Output:	William	Smith

There are three types of action you can perform: standardize, identify, and categorize.

If the term is found when performing the standardize action, Table Lookup replaces either the entire field or individual terms within the field with the standardized term, even if the field contains multiple words. Table Lookup can include changing full words to abbreviations, changing abbreviations to full words, changing nicknames to full names or misspellings to corrected spellings.

If the term is found when performing the identify action, Table Lookup flags the record as containing a term that can be standardized, but performs no action.

If the term is found when performing the categorize action, Table Lookup uses the source value as a key and copies the corresponding value from the table entry into the selected field. If none of the source terms match, **Categorize** uses the default value specified.

Input

Table 23: Table Lookup Input Fields

Field Name	Description / Valid Values
Source	Specifies the source input field to evaluate for scan and split.
StandardizationTable	One of the tables listed in Table Lookup Tables .

Options

Table Lookup options can be configured at the stage level, through any of the Spectrum Technology Platform clients, or at runtime, using dataflow options.

Configuring Options

To specify the options for Table Lookup you create a rule. You can create multiple rules then specify the order in which you want to apply the rules. To create a rule, open the Table Lookup stage and click **Add** then complete the following fields.

Note: If you add multiple Table Lookup rules, you can use the **Move Up** and **Move Down** buttons to change the order in which the rules are applied.

Option	Description
Action	<p>Specifies the type of action to take on the source field. One of the following:</p> <p>Standardize Changes the data in a field to match the standardized term found in the lookup table. If the field contains multiple terms, only the terms that are found in the lookup table are replaced with the standardized term. The other data in the field is not changed.</p> <p>Identify Flags the record as containing a term that can be standardized, but performs no action on the data in the field. The output field StandardizedTermIdentified is added to the record with a value of Yes if the field can be standardized and No if it cannot.</p> <p>Categorize Uses the Source value as a key and copies the corresponding value from the table into the field selected in the Destination list. This creates a new field in your data that can be used to categorize records.</p>

Option	Description
On	<p data-bbox="548 380 1422 436">Specifies whether to use the entire field as the lookup term or to search the lookup table for each term in the field. One of the following:</p> <p data-bbox="557 457 1422 804">Complete field Treats the entire field as one term, resulting in the following:</p> <ul data-bbox="691 506 1422 804" style="list-style-type: none"> <li data-bbox="691 506 1422 621">• If you selected the action Standardize, Table Lookup treats the entire field as one string and attempts to standardize the field using the string as a whole. For example, "International Business Machines" would be changed to "IBM". <li data-bbox="691 632 1422 709">• If you selected the action Identify, Table Lookup treats the entire field as one string and flags the record if the string as a whole can be standardized. <li data-bbox="691 720 1422 804">• If you selected the action Categorize, Table Lookup treats the entire field as one string and flags the record if the string as a whole can be categorized. <p data-bbox="557 835 1422 1276">Individual terms within field Treats each word in the field as its own term, resulting in the following:</p> <ul data-bbox="691 884 1422 1276" style="list-style-type: none"> <li data-bbox="691 884 1422 999">• If you selected the action Standardize, Table Lookup parses the field and attempts to standardize the individual terms within the field. For example, "Bill Mike Smith" would be changed to "William Michael Smith." <li data-bbox="691 1010 1422 1087">• If you selected the action Identify, Table Lookup parses the field and flags the record if any single term within the field can be standardized. <li data-bbox="691 1098 1422 1276">• If you selected the action Categorize, Unlike Standardize, Categorize does not copy the source term if there isn't a table match. If none of the source terms match, Categorize uses the default value specified. Unlike Standardize, Categorize only returns that table value and nothing from Source. If none of the source terms match, Categorize uses the default value specified.
Source	Specifies the field you want to containing the term you want to look up.
Destination	<p data-bbox="548 1482 1422 1530">Specifies the field to which the terms returned by the table lookup should be written.</p> <p data-bbox="548 1535 1422 1612">If you want to replace the value, specify the same field in the Destination field as you did in the Source field. You can also create a new field by typing the name of the field you want to create.</p> <p data-bbox="548 1629 1284 1656">The Destination field is not available if you select the action Identify.</p>
Table	<p data-bbox="548 1734 1422 1766">Specifies the table you want to use to find terms that match the data in your dataflow.</p> <p data-bbox="548 1780 1422 1835">For a list of tables that you can edit, see Table Lookup Tables. For information about creating or modifying tables, see Introduction to Lookup Tables.</p>

Option	Description				
Lookup multiple word terms	<p>Enables multiple word searches within a given string. For example:</p> <p>Input String: "Major General John Smith" Business Rule: Identify "Major General" in a string based on a table that contains the entry Output: Replace "Major General" with "Maj. Gen."</p> <p>For multiple word searches, the search stops at the first occurrence of a match.</p> <p>This option is disabled when On is set to Complete field.</p> <p>Note: Selecting this option may adversely affect performance.</p>				
When table entry not found, set Destination's value to	<p>Specifies the value to put in the destination field if a matching term cannot be found in the lookup table. One of the following:</p> <table border="0"> <tr> <td style="padding-right: 20px;">Source's value</td> <td>Put the value from the source field into the destination field.</td> </tr> <tr> <td>Other</td> <td>Put a specific value into the destination field.</td> </tr> </table>	Source's value	Put the value from the source field into the destination field.	Other	Put a specific value into the destination field.
Source's value	Put the value from the source field into the destination field.				
Other	Put a specific value into the destination field.				

Configuring Options at Runtime

Table Lookup options can be configured and passed at runtime if they are exposed as dataflow options. This enables you to override the existing configuration with JSON-formatted strings. You can also set stage options when calling the job through a process flow or through the job executor command-line tool.

You can find a schema for LookupRule in the following folder:

```
<Spectrum Location>\server\modules\jsonSchemas\tableLookup
```

To define Table Lookup rules at runtime:

1. In Enterprise Designer, open a dataflow that uses the Table Lookup stage.
2. Save and expose that dataflow.
3. Go to `Edit > Dataflow Options`.
4. In the **Map dataflow options to stages** table, expand Table Lookup. Check the box for LookupRule.
5. Optional: Change the name of the options in the **Option label** field.
6. Click **OK** twice.

Output

Table 24: Table Lookup Outputs

Field Name	Description / Valid Values
StandardizedTermIdentified	Indicates whether or not the field contains a term that can be standardized. Only output if you select Complete field or Individual terms in field options.
Yes	The record contains a term that can be standardized.
No	The record does not contain a term that can be standardized.

Transliterator

Transliterator converts a string between Latin and other scripts. For example:

Source	Transliteration
キャンパス	kyanpasu
Αλφαβητικός Κατάλογος	
биологическом	biologichyeskom

It is important to note that transliteration is not translation. Rather, transliteration is the conversion of letters from one script to another without translating the underlying words.

Note: Standard transliteration methods often do not follow the pronunciation rules of any particular language in the target script.

The Transliterator stage supports these scripts. In general, the Transliterator stage follows the UNGEGN Working Group on Romanization Systems guidelines. For more information, see www.eki.ee/wgrs.

Arabic The script used by several Asian and African languages, including Arabic, Persian, and Urdu.

Cyrillic	The script used by Eastern European and Asian languages, including Slavic languages such as Russian. The Transliterator stage generally follows ISO 9 for the base Cyrillic set.
Devanagari	The script used by several Indian languages, including Hindi and Sanskrit. This script is a descendent of the Brahmi script which is one of the oldest writing systems used in Ancient India and present South and Central Asia.
Greek	The script used by the Greek language. This script belongs to the Hellenic branch of the Indo-European language family.
Gujarati	The script used by the state of Gujarat in western India. It is one of the modern scripts of India which was adapted from the Devanagari script.
Gurmukhi	The script used by Indian language Punjabi. This script has a considerable influence from Nagari script which is an earlier form of the Devanagari script.
Hangul	The script used by the Korean language. The Transliterator stage follows the Korean Ministry of Culture and Tourism Transliteration regulations. For more information, see the website of The National Institute of the Korean Language .
Han	The script used by Chinese language. It is a branch of the Tibetan-Burman language family and has been written with scripts based on Thai and Chinese.
Traditional/Simplified Chinese	The Transliterator stage supports both traditional and simplified Chinese. For example, this is Traditional Chinese: 人人生而自由. This is Simplified Chinese: 人人生而自由
Kannada	The script used by several South Indian languages, such as Konkani. This script is a descendent of Brahmi script of ancient India.
Katakana and Hiragana	One of several scripts that can be used to write Japanese. The Transliterator stage uses a slight variant of the Hepburn system. With Hepburn system, both ZI (ジ) and DI (ヂ) are represented by "ji" and both ZU (ズ) and DU (ヅ) are represented by "zu". This is amended slightly for reversibility by using "dji" for DI and "dzu" for DU. The Katakana transliteration is reversible. Hiragana-Katakana transliteration is not completely reversible since there are several Katakana letters that do not have corresponding Hiragana equivalents. Also, the length mark is not used with Hiragana. The Hiragana-Latin transliteration is also not reversible since internally it is a combination of Katakana-Hiragana and Hiragana-Latin.
Half width/Full width	The Transliterator stage can convert between narrow half-width scripts and wider full-width scripts. For example, this is half-width: アルアノリウ. This is full-width: アルアノリウ.
Latin	The script used by most languages of Europe, such as English. It was originally used by the ancient Romans to write the Latin language.

Malayalam	The script used by the Malayalam language, the official language of the Indian state of Kerala. This script was first written with the Vatteluttu alphabet which means 'round writing' and developed from the Brahmi script of ancient India.
Oriya	The script used by the Oriya language, the official language of the Indian state of Odisha. The Oriya script was developed from the Kalinga script, one of the many descendents of the Brahmi script of ancient India.
Tamil	The script used by the Tamil language in several states of India, Sri Lanka, and Malaysia. This script was originally written with a version of the Brahmi script known as Tamil Brahmi.
Telugu	The script used by several languages of South India. This script is a descendent of Brahmi script of ancient India.
Thai	The script used by Thai language. This script is influenced by the Brahmi script of ancient India and the Khmer alphabets.

Transliterator is a part of Data Normalization. For a listing of other stages, see [Spectrum Data Normalization](#).

Transliteration Concepts

There are a number of generally desirable qualities for script transliterations. A good transliteration should be:

- Complete
- Predictable
- Pronounceable
- Unambiguous

These qualities are rarely satisfied simultaneously, so the Transliterator stage attempts to balance these requirements.

Complete

Every well-formed sequence of characters in the source script should transliterate to a sequence of characters from the target script.

Predictable

The letters themselves (without any knowledge of the languages written in that script) should be sufficient for the transliteration, based on a relatively small number of rules. This allows the transliteration to be performed mechanically.

Pronounceable

Transliteration is not as useful if the process simply maps the characters without any regard to their pronunciation. Simply mapping "αβγδεζηθ..." to "abcdefgh..." would yield strings that might be complete and unambiguous, but cannot be pronounced.

Standard transliteration methods often do not follow the pronunciation rules of any particular language in the target script. For example, the Japanese Hepburn system uses a "j" that has the English phonetic value (as opposed to French, German, or Spanish), but uses vowels that do not have the standard English sounds. A transliteration method might also require some special knowledge to have the correct pronunciation. For example, in the Japanese kunrei-siki system, "tu" is pronounced as "tsu". This is similar to situations where there are different languages within the same script. For example, knowing that the word Gewalt comes from German allows a knowledgeable reader to pronounce the "w" as a "v".

In some cases, transliteration may be heavily influenced by tradition. For example, the modern Greek letter beta (β) sounds like a "v", but a transform may continue to use a b (as in biology). In that case, the user would need to know that a "b" in the transliterated word corresponded to beta (β) and is to be pronounced as a "v" in modern Greek. Letters may also be transliterated differently according to their context to make the pronunciation more predictable. For example, since the Greek sequence GAMMA GAMMA ($\gamma\gamma$) is pronounced as "ng", the first GAMMA can be transcribed as an "n".

Note: In general, in order to produce predictable results when transliterating Latin script to other scripts, English text will not produce phonetic results. This is because the pronunciation of English cannot be predicted easily from the letters in a word. For example, grove, move, and love all end with "ove", but are pronounced very differently.

Unambiguous

It should always be possible to recover the text in the source script from the transliteration in the target script. For example, it should be possible to go from Elláda back to the original Ελλάδα. However, in transliteration multiple characters can produce ambiguities. For example, the Greek character PSI (ψ) maps to ps, but ps could also result from the sequence PI, SIGMA ($\pi\sigma$) since PI (π) maps to p and SIGMA (σ) maps to s.

To handle the problem of ambiguity, Transliterator uses an apostrophe to disambiguate character sequences. Using this procedure, the Greek character PI SIGMA ($\pi\sigma$) maps to p's. In Japanese, whenever an ambiguous sequence in the target script does not result from a single letter, the transform uses an apostrophe to disambiguate it. For example, it uses this procedure to distinguish between man'ichi and manichi.

Note: Some characters in a target script are not normally found outside of certain contexts. For example, the small Japanese "ya" character, as in "kya" (キヤ), is not normally found in isolation. To handle such characters, Transliterator uses a tilde. For example, the input "~ya" would produce an isolated small "ya". When transliterating to Greek, the input "a~s" would produce a non-final Greek sigma ($\alpha\sigma$) at the end of a word. Likewise, the input "~sa" would produce a final sigma in a non-final position ($\zeta\alpha$).

For the general script transforms, a common technique for reversibility is to use extra accents to distinguish between letters that may not be otherwise distinguished. For example, the following shows Greek text that is mapped to fully reversible Latin:


Input

Field Name	Description										
Any string field	The Transliterator stage can transliterate any string field. You can specify which fields to transliterate in the Transliterator stage options.										
TransliteratorID	<p>Overrides the default transliteration specified in the Transliterator stage options. Use this field if you want to specify a different transliteration for each record.</p> <p>For Example:</p> <table border="0"> <tr> <td>Arabic-Latin</td> <td>From Arabic to Latin.</td> </tr> <tr> <td>Greek-Latin</td> <td>From Greek to Latin.</td> </tr> <tr> <td>Latin-Hangul</td> <td>From Latin to Hangul.</td> </tr> <tr> <td>Latin-Katakana</td> <td>From Latin to Katakana.</td> </tr> <tr> <td>Fullwidth-Halfwidth</td> <td>From full width to half width.</td> </tr> </table>	Arabic-Latin	From Arabic to Latin.	Greek-Latin	From Greek to Latin.	Latin-Hangul	From Latin to Hangul.	Latin-Katakana	From Latin to Katakana.	Fullwidth-Halfwidth	From full width to half width.
Arabic-Latin	From Arabic to Latin.										
Greek-Latin	From Greek to Latin.										
Latin-Hangul	From Latin to Hangul.										
Latin-Katakana	From Latin to Katakana.										
Fullwidth-Halfwidth	From full width to half width.										

Options

Table 25: Transliterator Options

Option	Description/Valid Values
From	<p>The script used by the fields that you want to transliterate. For a description of the supported scripts, see Transliterator on page 238.</p> <p>Note: The Transliterator stage does not support transliteration between all scripts. The From and To fields automatically reflect the valid values based on your selection.</p>

Option	Description/Valid Values
To	<p>The script that you want to convert the field into. For a description of the supported scripts, see Transliterator on page 238.</p> <p>Note: The Transliterator stage does not support transliteration between all scripts. The From and To fields automatically reflect the valid values based on your selection.</p>
Swap button	<p>Click the swap button to exchange the languages in the From and To fields.</p> 
Fields to transliterate	Specifies the fields that you want to transliterate.
Remove Accent Marks	Select the check box to remove accent marks from the field. This option remains turned off by default.

Output

The Transliterator stage transliterates the fields you specify. It does not produce any other output.

Data Stewardship Stages

Introduction

Data Stewardship stages identify and manage exception records that Spectrum Technology Platform could not confidently process to allow manual review by a data steward.

Data Stewardship provides three stages in Enterprise Designer.

- **Exception Monitor**—A stage that evaluates records against a set of conditions to determine if the record requires manual review by a data steward. When records meet those conditions, this stage can send an email notifying recipients of the exceptions. An approval flow can be added to a condition to require acceptance by reviewers in addition to the data steward.
- **Write Exceptions**—A stage that writes the exception records to the exception repository. Once exception records are in the exception repository they are available for review by a data steward and (if there is an approval flow) acceptance by other reviewers.
- **Read Exceptions**—A stage that reads records from the exception repository into a dataflow. This stage allows you to reprocess exception records that have been corrected by a data steward.

Exception Monitor

The Exception Monitor stage evaluates records against a set of conditions to determine if the record requires manual review by a data steward. Exception Monitor enables you to route records that Spectrum Technology Platform could not successfully process to a manual review tool (the Data Stewardship Portal).

In the stage settings, you can set conditions that determine if records require manual review. You can configure notifications to email addresses when those conditions have been met a certain number of times. You can assign an approval flow type to a condition to create an approval flow.

For more information on exception processing, see [Data Stewardship Portal](#).

Input

Exception Monitor takes any record as input. If the input data does not contain a field called "CollectionNumber" the **Return all records in exception's group** option will be disabled.

Note: Exception Monitor cannot monitor fields that contain complex data such as lists or geometry objects.

Output

Exception Monitor returns records in two ports. The success port transmits records that do not meet any of the conditions defined in the Exception Monitor stage. The exception port transmits records that match one or more exception conditions. The exception port may also include non-exception records if you enable the option **Return all records in exception's group**. Exception Monitor only evaluates conditions. It does not modify fields within a record.

Reference

Conditions tab

This tab is displayed in the **Exception Monitor Options** dialog box.

Stop evaluating when a condition is met	Specifies whether to continue evaluating a record against the remaining conditions once a condition is met. Enabling this option may improve performance because it potentially reduces the number of evaluations that the system has to perform. However, if not all conditions are evaluated you will lose some degree of completeness in the exception reports shown in the Data Stewardship Portal. For example, if you define three conditions (Address Completeness, Name Confidence, and Geocode Confidence) and a record meets the criteria defined in Address Completeness, and you enable this option, the record would not be evaluated against Name Confidence and Geocode Confidence. If the record also qualifies as an exception because it matches the Name Confidence condition, this information would not be captured. Instead the record would be reported as having only an Address Completeness problem, instead of both an Address Completeness and Name Confidence problem.
--	---

Conditions list The list box on this tab shows all conditions that have been defined for the stage in tabular view. The conditions are evaluated in the order that they appear in the list. The list displays the following information for each condition.

- **Name**—The name for the condition. This is typically a meaningful name created by the user who created the condition.
- **Approval Flow Type**—The name of an approval flow type defined on the Management Console **Data Stewardship Settings** page.
- **Domain**—Specifies the kind of data being evaluated by the condition. This is used solely for reporting purposes to show which types of exceptions occur in the data.
- **Metric**—Specifies the metric that this condition measures. This is used solely for reporting purposes to show which types of exceptions occur in your data.
- **Assign to**—The user to whom the exception records meeting this condition are assigned. This user is referred to as the *data steward*. If this setting is empty, exception records from a job are automatically assigned to the user who ran the job.

Add Click this button to define a new condition.

Modify Click this button to edit the currently selected condition in the list.

Remove Click the button to remove the currently selected condition from the list.

Move Up/Move Down Click these buttons to order conditions that appear in the list. Conditions are evaluated in the order that they are displayed in the table. You can use these buttons to arrange selections when **Stop evaluating when a condition is met** is checked to assure that certain conditions more likely to be evaluated.

Configuration tab

This tab is displayed in the **Exception Monitor Options** dialog box.

Disable exception monitor Turns Exception Monitor on or off. If you disable Exception Monitor, records will simply pass through the stage and no action will be taken. This is similar in effect to removing Exception Monitor from the dataflow.

Stop job after reaching exception limit Specifies whether to halt job execution when the specified number of records meet the exception conditions.

Maximum number of exception records If **Stop job after reaching exception limit** is selected, use this field to specify the maximum number of exception records to allow before halting job execution. For example, if you specify 100, the job will stop once the 101st exception record is encountered.

Report only (do not create exceptions) Enables you to track records that meet exception conditions and reports those statistics on the Data Quality Performance page in the Data Stewardship Portal, but does not create exceptions for those records.

Return all records in exception's group Specifies whether to return all records belonging to an exception record's group instead of just the exception record. For example, a match group (based on a MatchKey) contains four records. One is the Suspect record, one is a duplicate that scored 90, and two are unique records that scored 80 and 83. If you have

a condition that says that any record with a MatchScore between 80 and 89 is an exception, by default just the records with a match score of 80 and 83 would be sent to the exception port. However, if you enable this option, all four records would be sent to the exception port.

Enable this option if you want data stewards to be able to compare the exception record to the other records in the group. By comparing all the records in the group, data stewards may be able to make more informed decisions about what to do with an exception record. For example, in a matching situation a data steward could see all candidates to determine if the exception is a duplicate of the others.

Note: If the input data does not contain a field named "CollectionNumber" this option will be disabled.

Group by If you selected **Return all records in exception's group**, choose the field by which to group the records.

Note: The "CollectionNumber" input field will not appear in this list because it is not a valid selection for the Group by feature.

Revalidation service Select the service you want to run when you revalidate records from this dataflow. The service runs when a user saves edited records in the Portal Exception Editor. Status is changed to Failed for records that fail revalidation. Successfully revalidated records are reprocessed or approved depending on the selection for **Action after revalidation**.

In an approval flow, successfully revalidated records are passed to the next approval level. For the last approval level in an approval flow, revalidated records are either released for reprocessing or retained in the repository as Resolved, depending on the selection for **Action after revalidation**.

Action after revalidation Specifies whether to reprocess records or approve records that have been successfully revalidated.

- **Reprocess records**—Choose this option to reprocess records that are successfully revalidated. The revalidated records are removed from the repository for reprocessing.
- **Approve records**—Choose this option to approve records that are successfully revalidated. The approved records are retained in the repository and their status changed to Resolved.

Match exception records using match field Uses match fields to match input records against exception records in the repository. Enable this option if your input contains records that previously generated exceptions but are now corrected in the input.

The input records will be evaluated against the conditions and then matched against the existing exception records in the repository. If an input record passes the conditions and matches an exception record, that exception record will be removed from the repository. If an input record does not pass the conditions and matches an exception record, that exception record will be updated and retained in the repository. Additionally, if duplicates exist in the

repository, only one matched exception per dataflow will be updated; all others for that dataflow will be deleted.

Optimized for single records or small batches

This option is activated when you check **Match exception records using match field**. When this option is not checked (default), the server will load into memory all existing exception records for the current dataflow and stage before processing the incoming exception records. This is recommended when the repository has a low number of existing exception records and high number of new exception records or updates. This scenario typically involves a longer initial load time and an increased memory requirement; it is faster when processing larger batches, such as daily, weekly, or monthly updates.

Checking this option is recommended when the repository has a high number of existing exception records and a relatively low number of new exception records or updates, as the server queries the repository for existing exception records as each input record is read in. This scenario typically involves a shorter initial load time and a lower memory requirement; it is faster when processing a few records in real time.



Match fields

Provides a list of all input fields used to build a key to match an exception record in the repository. You must define at least one match field if you checked the **Match exception records using match field** check box.

Add/Modify Condition dialog box

Conditions consist of one or more logical statements that evaluate the value in an input field. Use the **Add Condition** or **Modify Condition** dialog box to define the criteria to determine if a record is an exception.

The **Add Condition** dialog box is opened when you click **Add** on the **Conditions** tab of the **Exception Monitor Options** dialog box. It provides options to configure a new condition. The **Modify Condition** dialog box is opened when you click **Modify**. It allows you to edit values of the same options for an existing condition. After you have create conditions, they appear on the **Conditions** tab of the **Exception Monitor Options** dialog box.

Predefined Conditions Select a predefined condition or retain **<custom condition>** in the drop-down menu to create a new condition. After you have created predefined or custom conditions, they will appear on the **Conditions** tab of the **Exception Monitor Options** dialog box. The icon next to the name of the condition identifies it as either a predefined condition or a custom condition. A dual-document icon  designates a predefined condition, and a single document icon  designates a custom condition. Click the **Save** button to save a condition. After you save a condition, the **Predefined conditions** field changes to show the name of the condition rather than **<custom condition>**.

Name

A name for the condition. The name can be anything you like. Since the condition name is displayed in the Data Stewardship Portal, you should use a descriptive name. For example, `MatchScore<80` or `FailedDPV`. If you try to give a new condition a name that is identical to an existing condition but with other characters appended to the end (for example, `FailedDPV` and `FailedDPV2`), you will be asked whether you want to overwrite the existing condition as soon as you type the last character that matches its name (using our example, "V"). Respond **Yes** to the prompt, finish naming the condition, and when you press **OK** or **Save**, both conditions will be visible on the

Exception Monitor Options dialog box. The new condition will not overwrite the existing condition unless the new name is identical.

Assign to The user to whom the exception records meeting this condition are assigned. This user is referred to as the *data steward*. If this setting is empty, exception records from a job are automatically assigned to the user who ran the job. For an approval flow, you can configure this setting to specify a data steward other than the user who runs the job.

Condition categories

Data domain (Optional) Specifies the kind of data being evaluated by the condition. This is used solely 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", and so forth. You can specify your own data domain or select one of the predefined domains:

- 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.

Data quality metric (Optional) Specifies the metric that this condition measures. This is used solely 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. You can specify your own metric or select one of the predefined metrics:

- 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.

Approval flow

- Type** Specifies the name of an approval flow type. Approval flows define a succession of review levels through which exception records must be accepted by reviewers after they are edited by the data steward. When a condition is met, a record is associated with the approval flow specified here. The approval flow type is defined on the Management Console **Resources > Data Stewardship Settings** page. If the option is left as <undefined>, the condition will not be associated with an approval flow type, and records will be resolved by the data steward without any subsequent review.

Expression and Notification tabs

- Expressions** List expressions defined for a condition. Expressions are logical statements that check the value of a field to determine if the record might be considered an exception. Click the **Add** button to add a new expression. You must add at least one expression to a condition. Expressions are evaluated in the order that they appear here. You can click **Move Up** or **Move Down** to change the order.
- Notification** Complete options on this tab to send a message to email addresses when this condition is met a specific number of times. A notification email includes a link to the failed records in the Data Stewardship Portal **Editor**, where you can manually enter the correct data. If you do not wish to set up notifications, do not configure options on this tab. To stop sending notifications to a particular email address, remove that address from the list of recipients in the **Send notification to** box.
- **Send notification to**—Specifies email addresses for notifications. You can separate multiple email addresses with semicolons (;) or commas (,).
 - **Subject**—Specifies the subject line on email notifications.
 - **On**—Specifies to send a notification on the **First occurrence** that an expression evaluates to true or **After** a specified number of occurrences. The maximum allowed value is 1,000,000 occurrences.

- **Send reminder after**—Check this check box and specify the number of days on which to send a repeat email.
- **Remind daily**—Check this check box to continue sending reminders every day until the exception is resolved.
- **Reminder recipients**—Specifies recipients for reminders. The email addresses listed here do not have to match those specified for the original notification.

Add/Modify Expression dialog box

Use the **Add Expression** or **Modify Expression** dialog box to define expressions, which are logical statements that check the value of a field to determine if the record might be considered an exception. You must add at least one expression to a condition.

This dialog box opens when you click **Add** or **Modify** in the **Add/Modify Condition** dialog box.

Expression type

Expression created with Expression Builder	Select this option to create a basic expression.
Custom expression	Select this option to write an expression using Groovy scripting. If you need to use more complex logic, such as nested evaluations, use a custom expression. For more information, see Using Custom Expressions in Exception Monitor on page 252.

Expression

- Logical operator** If other expressions are already defined for this condition, you can select an operator to combine it with the preceding expressions.
- **And**—This expression must be true in addition to the preceding expression being true in order for the condition to be true.
 - **Or**—If this expression is true the condition is true even if the preceding expression is not true.

If you chose **Expression created with expression builder** for the **Expression type**, the following options are available.

Field name	Select the field that you want this expression to evaluate. The list of available fields is populated based on the stages upstream from the Exception Monitor stage.
Operator	Select the operator you want to use in the evaluation.
Value	Specify the value you want the expression to check for using the operator chosen in the Operator field. This is a regular expression if you selected matches regular expression as the Operator .

How to

Add the Exception Monitor stage to a workflow

1. In the **Stages** palette, expand **Primary Stages > deployed Stages > Data Stewardship**.
2. Drag the **Exception Monitor** stage to the canvas.
3. Optional: Click the Exception Monitor label and type a meaningful name for the stage in the workflow.
4. Drag a connection from an upstream stage to the input port on the Exception Monitor stage. This is the path through which records will enter the Exception Monitor.
5. Drag a connection from the success port to the first downstream stage that will process records that do not trigger any exceptions.
6. Drag a connection from the exception port to the first downstream stage that will process records that trigger an exception.
7. Double-click the Exception Monitor stage to configure its options.
8. On the **Conditions** tab, click the **Add** button to add conditions in the order that you want them evaluated.
 - a) In the **Add Condition** dialog box, complete **Name**, **Assign to**, **Data domain**, and **Data quality metric**. Under **Approval flow**.
For more information, see [Add/Modify Condition dialog box](#) on page 247.
 - b) Optional: In the **Type** box under **Approval flow**, you can select an approval type to trigger an approval flow with a condition.
To show on the **Conditions** tab, approval types must be defined on the Management Console **Data Stewardship Settings** page.
 - c) On the **Expressions** tab, click **Add** to add expressions to the condition.
You must add at least one expression to a condition.
For more information, see [Expression and Notification tabs](#) on page 249.
 - d) After you add expressions, click the **Move Up** or **Move Down** buttons to change the order in which expressions are evaluated.
 - e) Optional: On the **Notification** tab, you can add recipients for reminder emails.
For more information, see [Expression and Notification tabs](#) on page 249.
 - f) Click **OK**.
 - g) To save this condition for reuse as a predefined condition, click **Save**.
For an existing condition, you will be prompted whether to overwrite the condition.

Note: If you overwrite a predefined condition, any changes will affect all dataflows that already use the condition.
9. Use the **Move Up** and **Move Down** buttons to change the order in which conditions are evaluated. The order of the conditions is important only if you have enabled the option **Stop evaluating when a condition is met**. For more information, see [Conditions tab](#) on page 244

10. Configure options on the **Configuration** tab.
For more information, see **Configuration tab** on page 245.
11. Click the **OK** button.

Using Custom Expressions in Exception Monitor

You can write your own custom expressions to control how Exception Monitor routes records using the Groovy scripting language to create an expression.

Using Groovy Scripting

For information on Groovy, see groovy-lang.org.

Groovy expressions used in the Exception Monitor stage must evaluate to a Boolean value (true or false) that indicates whether the record is considered an exception and should be routed for manual review. Exception records are routed to the exception port.

For example, if you need to review records with a validation confidence level of <85, your script would look like:

```
data['Confidence'] < 85
```

The monitor would evaluate the value of the Confidence field against your criteria to determine which output port to send it to.

Checking a Field for a Single Value

This example evaluates to true if the Status field has 'F' in it. This would have to be an exact match, so 'f' would not evaluate to true.

```
return data['Status'] == 'F';
```

Checking a Field for Multiple Values

This example evaluates to true if the Status field has 'F' or 'f' in it.

```
boolean returnValue = false;
if (data['Status'] == 'F' || data['Status'] == 'f')
{
    returnValue = true;
}
return returnValue;
```

Evaluating Field Length

This example evaluates to true if the PostalCode field has more than 5 characters.

```
return data['PostalCode'].length() > 5;
```

Checking for a Character Within a Field Value

This example evaluates to true if the PostalCode field has a dash in it.

```
boolean returnValue = false;
if (data['PostalCode'].indexOf('-') != -1)
{
  returnValue = true;
}
return returnValue;
```

Common Mistakes

The following illustrate common mistakes when using scripting.

The following is incorrect because PostalCode (the column name) must be in single or double quotes

```
return data[PostalCode];
```

The following is incorrect because no column is specified

```
return data[];
```

The following is incorrect because row.set() does not return a Boolean value. It will always evaluate to false as well as change the PostalCode field to 88989.

```
return row.set('PostalCode', '88989');
```

Use a single equals sign to set the value of a field, and a double equals sign to check the value of a field.

Read Exceptions

Read Exceptions is a stage that reads records from the exception repository as input to a dataflow. (For more information on the exception repository, see [Data Stewardship Portal](#).)

Note: Once a record is read into a dataflow by Read Exceptions, it is deleted from the repository.

Input

The Read Exceptions stage reads data in from an exception repository. It does not take input from another stage in a dataflow.

Note: Only records marked as Resolved in the Data Stewardship Portal are read into the dataflow. This option can be changed by clearing the **Process resolved records** check box on the **Runtime** tab.

Output

The Read Exceptions stage has a primary Output port and an optional History port.

Output port

The primary Output port returns records from the Data Stewardship repository that have resolved status and that match the selection criteria specified in the Read Exception options. In addition to the record fields, Read Exceptions returns the following fields that describe the last modifications made to the record in the Data Stewardship Portal.

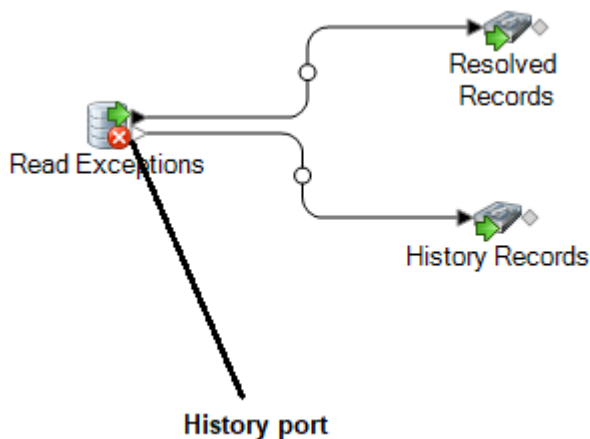
Table 26: Read Exceptions Output

Field Name	Description
Exception.Comment	Any comments entered by the person who resolved the exception. For example, comments might describe the modifications that the Data Stewardship made to the record.
Exception.LastModifiedBy	The last user to modify the record in the Data Stewardship Portal
Exception.LastModifiedMilliseconds	The time that the record was last modified in the Data Stewardship Portal. The time is expressed in milliseconds since January 1, 1970 0:00 GMT. This is the standard way of calculating time in the Java programming language. You can use this value to perform date comparisons, or to create a transform to convert this value to whatever date format you want.

Field Name	Description
Exception.LastModifiedString	The time that the record was last modified in the Data Stewardship Portal. This field provides a more understandable representation of the date than the Exception.LastModifiedMilliseconds field. The time is expressed in this format: Thu Feb 17 13:34:32 CST 2011

History port

The optional history port streams all versions of records. History records are typically used to assemble change logs.



Options

Settings in the Read Exceptions stage are configured on the **General**, **Sort**, and **Runtime** tabs.

General Tab

The options on the **General** tab specify which exception records you want to read into the dataflow.

The **Filter** options allow you to select a subset of records from the exception repository using these criteria.

- User** The user who ran the dataflow that generated the exceptions you want to read into the dataflow.
- Dataflow name** The name of the dataflow that generated the exceptions you want to read into the dataflow.
- Stage label** The Exception Monitor stage's label as shown in the dataflow in Enterprise Designer. This criteria is useful if the dataflow that generated the exceptions

contains multiple Exception Monitor stages and you only want to read in the exceptions from one of those Exception Monitor stages.

From date	The date and time of the oldest records that you want to read into the dataflow. The date of an exception record is the date it was last modified.
To date	The date and time of the newest records that you want to read into the dataflow. The date of an exception record is the date it was last modified.

The **Fields** listing shows the fields that will be read into the dataflow. By default all fields are included, but you can exclude fields by clearing the check box in the **Include** column.

The **Preview** listing shows the records that meet the criteria you specified under **Filter**.

Note: The preview displays only records that have been marked "Approved" in the Data Stewardship Portal and meet the filter criteria.

Sort Tab

Options on the **Sort** tab specify how to sort input records based on field values.

Add	Adds a field to sort on.
Field Name column	Shows the name of the field to sort on. You can select a field by clicking the drop-down button.
Order column	Specifies whether to sort in ascending or descending order.
Up and Down	Changes the order of the sort. Records are sorted first by the field at the top of the list, then by the second, and so forth.
Remove	Removes a sort field.

Runtime Tab

Options on the **Runtime** tab specify runtime behavior of the Read Exceptions stage.

Starting record	Specify the position in the repository of the first record you want to read into the dataflow. For example, if you want to skip the first 99 records in the repository, you would specify 100. The 100th record would be the first one read into the repository if it matches the criteria specified on the General tab. A record's position is determined by the order of the records in the Data Stewardship Portal.
All records	Select this option if you want to read in all records that match the search criteria specified on the General tab.
Max records	Select this option if you want to limit the number of records read in to the dataflow. For example, if you only want to read in the first 1,000 records that match the selection criteria, select this option and specify 1000.
Process resolved records	Select this check box to process resolved records. Clear this check box to process and purge unresolved records. By default, this check box is selected.

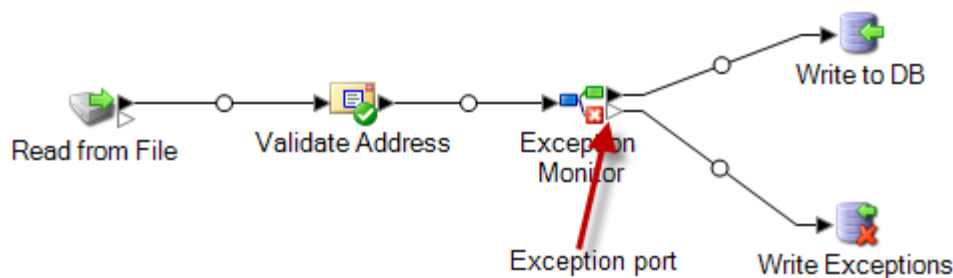
Write Exceptions

Write Exceptions is a stage that takes records that the Exception Monitor stage has identified as exceptions and writes them to the exception repository. Once in the exception repository, the records can be reviewed and edited using the Data Stewardship Portal.

Note: Exception records are not written to the Data Stewardship Portal when jobs or services are run in inspection mode in Enterprise Designer or preview mode in Management Console.

Input

The Write Exceptions stage takes records from the exception port on the Exception Monitor stage and then writes them to the exception repository. The Write Exceptions stage should be placed downstream of the Exception Monitor stage's exception port. The exception port is the bottom output port on the Exception Monitor stage:



Output

Write Exceptions does not return any output in the dataflow. It writes exception records to the exception repository.

Options

The Write Exceptions stage enables you to select which fields' data should be returned to the exceptions repository. The fields that appear depend upon the stages that occur upstream in the dataflow. If, for instance, you have a Validate Address stage in the dataflow, you would see such fields as AddressLine1, AddressLine2, City, PostalCode, and so on in the Write Exceptions stage. By default, all of those fields are selected; uncheck the boxes for any fields you do not want returned to the exceptions repository. The order of the fields is determined by how they are ordered when they come into the Write Exceptions stage. You can reorder the fields by selecting a row and using the arrows on the right side of the screen to move the row up or down. The order you select here will persist for all users in the Data Stewardship Portal, but each user can reorder the fields within the Portal to their own liking.

Select **Allow user to create best of breed records in Portal** to perform manual record consolidation when data matches cannot be made using standard rules. This feature copies a selected record within a group and uses it instead of the duplicates for processing. This option is available only for

grouped exception records that are generated from a matching job, which is identified by the presence of the CandidateGroup field or the CollectionNumber field. When you use this option, a read-only field called "CollectionRecordType" will be added to the exception record. You can see this field at the bottom of the list; note that all options for that field are disabled.

Note: If your dataflow is also being configured for **revalidation**, you will need to manually add and expose the CollectionRecordType field in the Exception Monitor stage/subflow and the service itself.

After adding a **Best of Breed** record in the Resolve Duplicates view of the Data Stewardship Portal **Editor**, this field will be set to "BestOfBreed." If you choose to create best of breed records, you will be unable to use the **Approve All** option for those records in the Data Stewardship Portal. Read more about best of breed records in the Data Stewardship Portal [here](#).

You may have input fields that you want in the repository but do not want to be viewable within the Data Stewardship Portal. This could be due to the field containing sensitive data or simply because you want to streamline what appears in the Portal. Check the **Allow viewing** box to designate which of the selected fields should be viewable once they are passed to the exceptions repository. By default, all fields are viewable. Uncheck the box for any field you do not want visible in the Portal.

Additionally, you can designate which of the selected fields should be editable in the Portal once they are passed to the exceptions repository. By default, the **Allow editing** column is checked for all fields coming in to the Write Exceptions stage. Uncheck the box for any field you wish to be returned to the exceptions repository in a read-only state.

Finally, you can use the **Lookup** function to assign a lookup to a field containing problematic data. You can select from the list of lookups that have been defined in the Data Stewardship Settings tool or you can manually enter the name of the lookup. For more information on lookups, see [Lookups](#).

Note: Lookups can be assigned only to fields whose type is string.

Enterprise Data Integration Stages

Call Stored Procedure

Call Stored Procedure is a source stage that executes a stored procedure in a database, and returns the results of the stored procedure call as input for the dataflow. Use Call Stored Procedure when you want to get data from a database using a database's stored procedure rather than a query to a table or view.

Note: If you want to read data into a dataflow directly from a table or view, use the Read from DB stage.

You may want to use Call Stored Procedure to read data into a dataflow if you have business logic embedded in the stored procedure and you want to use that logic in your Spectrum Technology

Platform environment. For example, many operational systems do not use referential integrity checks in the database for large constantly-updated tables because of the reduction in performance that such checks would cause. So to maintain referential integrity, you could create stored procedures and use them for all updates to the system.

Stored procedures can also be used to simplify management of the Spectrum Technology Platform environment. For example, if you have hundreds of ETL processes that all read the same data, you may want to put the query into a stored procedure so it is in one place. This makes maintenance easier since you only need to modify the one stored procedure instead of hundreds of different processes.

Option Name	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Spectrum Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage Connections.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection name Enter a name for the connection. The name can be anything you choose.</p> <p>Database driver Select the appropriate database type.</p> <p>Connection options Specify the host, port, instance, user name, and password to use to connect to the database.</p>
Schema	Specifies the schema that contains the stored procedure you want to call.
Procedure	Specifies the stored procedure you want to call.

Option Name	Description
Stored Procedure Parameters	<p data-bbox="548 380 1263 405">This table specifies the values for the stored procedure parameters.</p> <p data-bbox="557 426 1419 451">Parameters This column shows the parameters defined in the stored procedure.</p> <p data-bbox="557 472 1419 657">Stage Fields For OUT, INOUT, and RETURN parameters, this column shows the dataflow field name that will contain the data returned by the parameter. Initially, the field name is the same as the parameter name. You can modify the stage field name by clicking the field name and typing a new name for parameters. This column is not used for IN parameters.</p> <p data-bbox="557 678 1419 1119">Direction One of the following:</p> <ul style="list-style-type: none"> <li data-bbox="711 720 1419 814">IN The parameter is an input parameter. The value you specify for this parameter is passed to the stored procedure as input. <li data-bbox="711 835 1419 898">OUT The parameter is an output parameter. The stored procedure returns data to the stage in this parameter. <li data-bbox="711 919 1419 1035">INOUT The parameter can be used as both an input parameter to pass a value to the stored procedure, and as an output parameter to receive data returned by the stored procedure. <li data-bbox="711 1056 1419 1119">RETURN The parameter contains a return code from the stored procedure. <p data-bbox="557 1150 1419 1266">Types This column displays the data type of the parameter value. If the data type is not supported by Spectrum Technology Platform, the type will be "Unsupported" and the stored procedure will not execute successfully.</p> <p data-bbox="557 1287 1419 1350">Value In this column, enter the value you want to set for the parameter. This column is disabled for OUT parameters.</p>

Option Name	Description
Result Set Fields	<p>This table specifies which dataflow fields to use for the data returned by the stored procedure.</p> <p>Database Tables This column shows the tables from which the stored procedure returned data.</p> <p>Database Fields This column shows the field from which the stored procedure returned data.</p> <p>Stage Fields This column shows the dataflow field name that will contain the data from the database field.</p> <p>Types This column shows the data type of the field. If the data type is not supported by Spectrum Technology Platform, the type will be "Unsupported".</p> <p>Include Check the box in this column to include the field in the dataflow. If the box is not checked, the field will not be used in the dataflow.</p>
Get Fields	Click this button to populate the Result Set Fields table with the result set schema returned by the stored procedure. This will execute the stored procedure and get the result set schema.
Add	Click this button to add a result set field manually.
Remove	Click this button to remove a result set field from the list of available fields.

DB Change Data Reader

The **DB Change Data Reader** stage allows you to select the columns to be included in the current jobflow, where the columns have the Change Data Capture feature enabled on them.

In the stage, you can create a Change Data Capture (CDC) Resource, which is the required data source table. The columns of the CDC resource on which the Change Data Capture feature is enabled are reflected using checkboxes.

Change Data Capture

The Change Data Capture feature enables capture of all changes made in a column. For each selected column, all inserts, updates, and deletions are captured.

Supported Databases

Currently, Spectrum Technology Platform supports the Change Data Capture (CDC) feature for MS SQL and Oracle databases only.

MS SQL For MS SQL data sources, the Change Data Capture feature can be enabled or disabled for columns of tables from the backend. Refer to [here](#) for the necessary steps.

Note: The Change Data Capture feature is not supported in the Express edition of SQL Server.

Oracle For Oracle data sources, Spectrum Technology Platform tracks the data changes in the table columns using the LogMiner utility of Oracle. CDC cannot be enabled or disabled on columns of tables in an Oracle data source through Spectrum Technology Platform.

The data changes resulting from `insert`, `update`, and `delete` queries are tracked and captured from an entered start date and time up to the current date and time. This entered start time is applicable during the first query execution with CDC switched on.

In subsequent executions on the same Oracle connection, data changes are captured from the time of the last execution till the current time.

Note: This is an incremental process. In the first capture, the changes are captured from the entered start date and time to the current date and time. In subsequent captures, the changes are captured from after the end date and time of the previous capture to the current date and time.

Adding a CDC Resource

Note: To use the Change Data Capture feature, ensure the SQL Server Agent is running on the MS SQL server.

1. Open the **Change Data Capture Management** popup, through either of the below two ways:
 - Navigate to **Tools > Change Data Capture Management**.
 - Add the **DB Change Data Reader** stage to a job, open the stage settings, and click **Manage**.
2. Click **Add**.
3. Enter a **Name** for the CDC Resource.
4. In the **Connection** field, select the SQL database connection you want to use. To make a new database connection, click **Manage**. For more information on creating database connections, see [Database Connection Manager](#) on page 397.
5. In the **Table/View** field, specify the table whose columns are to be included in the jobflow. Click the browse button ([...]) to navigate to the table or view that you want to use. The grid below displays all the columns of the selected table, along with the datatypes of each column.
6. If you select an Oracle connection in the **Connection** field, the **Start date** field becomes available.

The field is filled with the default value of the current date with the time 12:00 AM. You can enter a start date and time of your choice.

The end date and time is taken as the current date and time.

Attention: You should have *execution* rights on the Oracle LogMiner utility to be able to use the CDC feature on an Oracle connection. For more information, see [Oracle LogMiner Configurations](#).

Note: The **Start date** field is not available on selecting an MS SQL connection in the **Connection** field.

7. Click **OK**.

The created CDC Resource table is now ready for use in the **DB Change Data Reader** stage, whose columns can be included or excluded from the jobflow.

The stage displays the table columns on which the CDC feature has been enabled.

Editing a CDC Resource

1. Open the **Change Data Capture Management** popup, through either of the below two ways:
 - Navigate to **Tools > Change Data Capture Management**.
 - Add the **DB Change Data Reader** stage to a job, open the stage settings, and click **Manage**.
2. Select the CDC Resource you need to modify.
3. Click **Edit**.
4. Modify the details of the added CDC Resource as required.
5. Click **OK**.

Deleting a CDC Resource

1. Open the **Change Data Capture Management** popup, through either of the below two ways:
 - Navigate to **Tools > Change Data Capture Management**.
 - Add the **DB Change Data Reader** stage to a job, open the stage settings, and click **Manage**.
2. Select the CDC Resource you need to delete.
3. Click **Delete**.

Selecting Change Data Reader Options

The **DB Change Data Reader Options** reflects the table columns of the selected CDC Resource for which the CDC feature is enabled.

You can select which columns to include or exclude in the current jobflow.

1. In a job, add the **DB Change Data Reader** stage.
2. Open the **DB Change Data Reader Options** by double-clicking the stage icon.

3. Select the desired CDC Resource from the **Select a resource** dropdown.

You can add or modify a CDC Resource by clicking **Manage**.

The grid below displays all the table columns with their data types. It also displays whether a particular column is included in the job flow, and whether the column is selected for the Change Data Capture feature.

4. Using the check boxes under the **Include** column of the grid, select the table columns to be included in the job flow.
5. The check boxes under the **CDC Enabled** column of the grid reflect the table columns on which the CDC feature is enabled.

The read-only **CDC Enabled** check boxes are checked for columns on which the CDC feature is enabled.

Note: For MS SQL data sources, refer [here](#) for the steps to enable or disable the Change Data Capture feature on particular table columns from the back end.

6. Click **OK**.

The data of the table columns, which have been selected for Change Data Capture, is captured and saved.

DB Loader

The **DB Loader** stage allows you to access and load data from/to databases configured in Enterprise Data Integration. This stage provides an interface to a high-speed data loading utility. Currently, the Enterprise Data Integration platform supports **Oracle Loader**, **DB2 Loader**, **PostgreSQL Loader**, and **Teradata Loader**.

Oracle Loader

The Oracle Loader allows you to load data to any Oracle database configured in the Enterprise Data Integration platform.

Note: Oracle client must be installed with administrator setup before using the Oracle Loader.

Option Name	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Spectrum Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage Connections.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection name Enter a name for the connection. The name can be anything you choose.</p> <p>Database driver Select the appropriate database type.</p> <p>Connection options Specify the host, port, instance, user name, and password to use to connect to the database.</p>
Table/View	<p>After selecting a connection, specify the table or view to write to. Click the browse button ([...]) to go to the table or view that you want to use, or click Create Table to create a new table in the database.</p>
Listener	<p>Specify a variable name that contains the address and connection details required to establish a connection to the Oracle database. For example, "XE". This variable is present in <code>tnsnames.ora</code>, a file that contains client side network configuration parameters.</p>
Stage fields	<p>This column lists the field names used in the dataflow. You cannot modify these field names.</p>
Types	<p>This column lists the data type of each field.</p>

Runtime Tab

Option Name	Description
Load method	<p>Specifies how you want to write data into the Oracle database.</p> <p>Append Adds the data into the target table without erasing the table's existing data.</p> <p>Insert Loads the data into the database. The table must be empty before it is loaded. It doesn't work on multiple runtime instances.</p> <p>Truncate and Insert Deletes existing rows if any, then loads the input data into the table. This does not work on multiple runtime instances.</p>
Use direct path loader	<p>Select this to load data directly into the Oracle database bypassing much of the data processing that normally takes place.</p>

Option Name	Description
Unrecoverable	This check box is enabled when you select Use direct path loader . Select this if you do not want to write Redo logs in the database. For more information about Redo logs, see http://docs.oracle.com/cd/B28359_01/server.111/b28310/onlineredo001.htm#ADMIN11302
Log file folder	Specifies the path to the folder. Click the ellipses button (...) to browse to the folder you want. The log file contains a record of this stage's activities during a load session.
Bad file folder	Specifies the path to the folder. Click the ellipses button (...) to browse to the folder you want. The bad file contains a list of records that the stage fails to load into the database.
Maximum errors allowed	Specify the maximum number of errors to allow before halting the load operation. To halt the load operation on the first error, set value to 0. A maximum of 32767 errors are allowed.

Note: You can achieve significant performance improvements by using multiple runtime instances of this operation. To do this, click the **Runtime** button and enter the required value in the **Runtime instances** field.

DB2 Loader

The DB2 Loader allows you to load data to any DB2 database configured in the Enterprise Data Integration platform. You need to set up the DB2 Utility on the same machine where you are running the Spectrum server.

Perform these steps:

1. Install the DB2 runtime client with Administrator set up.
2. Configure the loader utility as described in the table below.
3. Start the Spectrum server.

Note: In case the Spectrum server was already running when you started the configuration, you will need to restart the server for the configuration to take effect.

Option Name	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Spectrum Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage Connections.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection name Enter a name for the connection. The name can be anything you choose.</p> <p>Database driver Select the appropriate database type.</p> <p>Connection options Specify the host, port, instance, user name, and password to use to connect to the database.</p>
Table/View	<p>After selecting a connection, specify the table or view to write to. Click the browse button ([...]) to go to the table or view that you want to use, or click Create Table to create a new table in the database.</p>
Database/Alias	<p>This is a variable that catalogues the DB2 server and database.</p> <p>To catalog the DB2 server Use the DB2 command line processor on spectrum server machine and enter the command:</p> <pre>CATALOG TCPIP NODE <nodename> REMOTE <hostname> SERVER <port></pre> <p>where:</p> <p>nodename: name of connection</p> <p>hostname: TCP/IP name of the DB2 server machine</p> <p>port: server port</p> <p>To catalog the database Use the command:</p> <pre>CATALOG DATABASE <databasename> AS <local_database_alias> AT NODE <nodename></pre> <p>where:</p> <p>databasename: Name of the database on the DB2 server</p> <p>local_database_alias: Local Name given to the database while connecting from the server machine</p> <p>nodename: Name used in the previous CATALOG TCP/IP command</p>
Stage fields	<p>This column lists the field names used in the dataflow. You cannot modify these field names.</p>
Types	<p>This column lists the data type of each field.</p>

Runtime Tab

Option Name	Description
Load method	<p>Indicates the mode of writing data into a DB2 table.</p> <p>Insert Inserts the loaded data into the table, while the existing table data remains unchanged.</p> <p>Replace Inserts the loaded data into the table after deleting all existing data from it.</p> <p> The table schema and index definitions remain unchanged.</p> <p>Restart Restarts the data load, in case the previous load attempt was interrupted.</p>
Non-recoverable	<p>Indicates if this load transaction is non-recoverable.</p> <p>If you select this option, the load transaction is marked as non-recoverable. Table spaces are not put into the Backup Pending state after the load, nor is a copy of the loaded data made during the load. Hence, a non-recoverable transaction cannot be recovered in the event of a data load failure, even if a <code>rollforward</code> is attempted later.</p> <p>If you select this option, you cannot recover from the transaction even if you use the DB2 <code>rollforward</code> utility because the utility skips such a non-recoverable transaction, and the table is marked as "invalid". In addition, subsequent transactions against the table are also ignored by <code>rollforward</code>.</p> <p>To restore a table that contains non-recoverable transactions, you must use either a tablespace-level backup or a full backup taken at a commit point following the non-recoverable load.</p> <p>Note: Do not select this option if the data contains Datalink columns that have the File Link Control attribute present in them.</p>
CPU	The number of parallel threads that the load utility can generate and sustain for loading, parsing and formatting the records, while building table objects in each database partition.
Disk	The number of parallel threads that the load utility can generate and sustain for writing data to table space containers.

Option Name	Description
Indexing Mode	<p>Indicates the mode of handling of indexes by the load utility.</p> <p>Autoselect The load utility decides whether to apply Rebuild or Incremental mode, based on the amount of data and the depth of the index tree.</p> <p>Rebuild All indexes are rebuilt.</p> <p>Incremental New data is added to the existing indexes.</p> <p>This mode can be applied only if the index object is valid and accessible at the start of a load operation.</p> <p>Note: Incremental indexing is not supported if ALL of these conditions hold true:</p> <ol style="list-style-type: none"> 1. The Load Copy option is specified (<code>logretain</code> or <code>userexit</code> is enabled). 2. The table resides in a DMS table space. 3. The index object resides in a table space that is shared by other table objects belonging to the table being loaded. <p>To bypass this limitation, place indexes in separate table spaces.</p> <p>Deferred The load utility does not attempt creating an index. Existing indexes are marked for refresh.</p> <p>Note: Index construction requires more time in Deferred mode than in Rebuild mode. Hence, while performing multiple load operations, allow the last load operation to rebuild all indexes instead of rebuilding indexes at the first access by a non-load operation.</p> <p>Note: This mode is supported only for tables with non-unique indexes.</p>
Fast Parse	<p>Indicates whether syntactical validation on column values must be left out, thus enhancing performance.</p> <p>If checked, any syntactical errors in the data are ignored in favor of optimized performance.</p> <p>For example, if a String value <code>12wxvvg56</code> is encountered in a field mapped to an integer column in an ASCII file, the load utility should normally generate a syntax error. But if Fast Parse is selected, the syntax error is ignored, and a random number is loaded into the integer field.</p> <p>Note: Ensure you use this option only with correct and clean data.</p>
Schema Name	The schema in which the exception tables are stored.
Table Name	The exception table into which those rows are copied in which some error is encountered while loading.

Option Name	Description
Log file folder	<p>The path of the directory in which the log files are to be stored.</p> <p>A log file contains a list of the database load transactions run by a DB Loader stage in one load session.</p> <p>Click the ellipses button (...) to specify the desired directory for log files.</p>
Bad file folder	<p>The path of the directory on the DB2 server in which the bad files are to be stored.</p> <p>A bad file contains a list of the records that a DB Loader stage fails to load into the database.</p> <p>Click the ellipses button (...) to specify the desired directory for bad files.</p>
Maximum errors allowed	<p>The maximum number of errors allowed before a load operation is aborted.</p> <p>To abort a load operation as soon as the first error is encountered, set the value of this field to 0.</p> <p>Note: A maximum of 32767 errors are allowed.</p>
Parallelism	<p>A DB2 database can be divided into multiple partitions by cloning the environment onto different physical nodes.</p> <p>Separate database requests for data fetch and update are automatically divided amongst the different partitions and run in parallel for optimized performance.</p>
Exception Handling	<p>A DB2 database allows you to record the errors and exceptions encountered while running queries and procedures, and also handle them appropriately.</p> <p>For this, a DB2 database provides specific exception tables and schema which store the source as well as the log traces of each database exception.</p> <p>To use the Exception table, ensure:</p> <ul style="list-style-type: none"> • DB2 runtime client is version 10.5 or above • Service pack version is 7 or above

PostgreSQL Loader

The PostgreSQL Loader allows you to load data to any PostgreSQL database configured in the Enterprise Data Integration platform.

Option Name	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Spectrum Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage Connections.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection name Enter a name for the connection. The name can be anything you choose.</p> <p>Database driver Select the appropriate database type.</p> <p>Connection options Specify the host, port, instance, user name, and password to use to connect to the database.</p>
Table/View	Click the browse button ([...]) to navigate to the table or view you want to use.
Database fields	This column lists the field name in the database. You cannot modify these field names.
Stage fields	This column lists the field names used in the dataflow. You cannot modify these field names.
Types	This column lists the data type of each field.

Runtime Tab

Option Name	Description
Load method	<p>Specifies the method of writing data to the PostgreSQL database tables.</p> <ul style="list-style-type: none"> • Select Insert to write data to an empty table or to append it to an existing data table. • Select Truncate and Insert to truncate the data before loading it to the database table.

Note: You can achieve significant performance improvements by using multiple runtime instances of this operation. To do this, click the **Runtime** button and enter the required value in the **Runtime instances** field.

Teradata Loader

The Teradata Loader allows you to load data to any Teradata database configured in the Enterprise Data Integration platform.

Note: The loader is supported only on Windows systems.

Option Name	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Spectrum Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage Connections.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection name Enter a name for the connection. The name can be anything you choose.</p> <p>Database driver Select the appropriate database type.</p> <p>Connection options Specify the host, port, instance, user name, and password to use to connect to the database.</p>
Table/View	Click the browse button ([...]) to navigate to the table or view you want to use.
Database fields	This column lists the field names in the database. You cannot modify these field names.
Stage fields	This column lists the field names used in the dataflow. You cannot modify these field names.
Types	This column lists the data type of each field.

Runtime Tab

Option Name	Description
Load method	<p>Specifies the method of writing data to the Teradata database tables.</p> <ul style="list-style-type: none"> • Select Insert to write data to an empty table • Select Append to write data to an existing data table. • Select Truncate and Insert to truncate the data before writing it to the database table.
Log file folder	Select the folder location for saving the log files of the upload process.
Generate bad file	Mark this check-box to generate a log of the records that failed to get uploaded.

Option Name	Description
Bad file folder	Select the folder location for saving the log of failed records. The log gives details, such as error codes and error field names of the failed records.
Maximum errors allowed	Specify the limit of permitted errors for the upload session. If the number of errors exceeds this value, the upload process pauses.

Note: Multiple runtime instances are not supported for Teradata Loader.

Field Parser

The **Field Parser** stage extracts fields from XML and delimited data in the specified input column. To configure the **Field Parser** options, perform the following tasks.

1. From the **Source** field select the column that has the XML or delimited data to be parsed.

Note: The drop-down displays all the string input columns.

2. Select the `XML` or `Delimited` **Format** based on the type of data you want to parse, and accordingly, select the options described below.

Field Parser Options for XML Data

Option Name	Description
Server name	Indicates whether the file selected for inferring the schema is located on the computer running the Spectrum Enterprise Designer or on the server. If you select a file on the local computer, the server name will be My Computer. If you select a file on the server the server name will be Spectrum Technology Platform.

Option Name	Description
Schema file	<p>Specifies the path to an XSD schema file. Click the ellipses button (...) to navigate to the file location. The schema file can reside on the server or your local system.</p> <p>Alternatively, you can also specify an XML file instead of an XSD file. If you specify an XML file the schema will be inferred based on the structure of the XML file. Using an XML file instead of an XSD file has the following limitations:</p> <ul style="list-style-type: none"> • The XML file cannot be larger than 1 MB. If the XML file is more than 1 MB in size, try removing some of the data while maintaining the structure of the XML. • The data file will not be validated against the inferred schema. <p>Note: If the Spectrum Technology Platform server is running on Linux, remember that file names and paths on these platforms are case sensitive.</p>
Output Fields	<p>This section displays details of the selected schema. It includes the root element followed by the child elements along with their attributes.</p> <p>By default all the nodes of the schema remain selected. However, you can clear the check-box of the nodes that you do not want to be passed to the next stage.</p> <ul style="list-style-type: none"> • Search node: Type the name of the node to which you want to navigate in the schema tree. The typed node gets highlighted in the preview pane below the field. • XPath: Click anywhere in this field to view the XML path (XPath) of the elements and attributes of the highlighted node in schema tree. To see all the previous XPaths viewed by you, click the down arrow at the right end of the field. <p>Note: XPath is a language for finding information in an XML document. For further details on this, see https://www.w3schools.com/xml/xml_xpath.asp</p>

Field Parser Options for Delimited Data

Option Name	Description
Field separator	<p>From the dropdown list, select the field separator used in the delimited column to be parsed.</p> <p>If the delimited column uses a different character as a field separator, click the ellipses button to select another character as field separator.</p>
Text qualifier	<p>From the dropdown list, select the text qualifier used in the delimited column to be parsed.</p> <p>Note: Text qualifiers are the character used to surround text values in a delimited data.</p> <p>If the delimited column uses a different text qualifier, click the ellipses button to select another character as a text qualifier.</p>

Option Name	Description
Output type	<p>Select if you want the parsed output in the form of a List (hierarchical display of values) or Fields.</p> <p>Note: For list as the output type, you can add only one output field, whereas the <code>Fields</code> option allows you to add multiple fields in which you can get the values segregated during parsing.</p>
Output Fields	<p>This section allows you to add/modify the various fields in which you want details of the delimited column to be segregated. You can also delete any of the added output fields.</p> <p>To add a new field for displaying the parsed output, click the Add button, and perform these steps in the Field Setting pop-up that is displayed:</p> <ol style="list-style-type: none"> Enter the Name of the field. From the Type drop-down, select the data type for the field being added. Based on the selected type, few more fields can be defined. For example, in case of <code>date</code>, you can define its format as <code>M/d/yy</code>, <code>MMM d.yyyy</code>, or <code>MMMM d.yyyy</code>. For details on the data types and defining its details, see Defining Fields In a Delimited Input File on page 312. <p>Note: If you select <code>String</code> as the data type, any type of delimited data will be parsed. However, you can also use the specific type, based on the data you want to parse in the field.</p> <ol style="list-style-type: none"> In the Position field, enter the position of the data type (in the input file) that is to be parsed to this field. For example, in the following file snippet, if you want to parse the date time values to the field being added, enter the Position as 3. <pre> true;"02/02/2022";"10/2/92 5:05 AM";598985994665542.25634;1; "Arjun";74785.155;5:05PM,1,Deepak,65152 false;"15/03/1923";"3/23/90 11:55 AM";3425699466554.2563;2; "sharma";5.1;5:45AM,2,Arjun,365273 </pre> <ol style="list-style-type: none"> Click Add Field and Close. <p>The added field and its details are displayed in the box.</p> <p>Note: If you want to have any excess space characters removed from the beginning and end of a field's value string, select the Trim check box.</p> <p>Modify: Click this button to modify details of any of the added output fields.</p> <p>Remove: Click this button to delete any of the added output fields.</p>

Runtime: Use this button to specify multiple runtime instances of parser. This results in significant performance improvement.

OK: Click this button to save all the details entered in this stage.

Cancel: Click this button to cancel all the updates you made.

Help: Click this button to refer to the help file for this stage.

Field Combiner

The **Field Combiner** stage combines fields coming from the previous stage in a dataflow to create an XML string.

Field Combiner Options

Option Name	Description
Output column name	Specify the name of the column that will have the combined fields as XML string.
Output Fields	<p>This section helps you select the fields to be combined and perform various actions on those fields.</p> <p>Note: The Apply Namespace dropdown list and Element, Attribute, and Remove buttons get enabled when you select one or more elements/attributes in the schema pane.</p> <ul style="list-style-type: none"> • Quick Add: Click this button to open the Add/Remove Fields pop-up window. This window displays a list of all the fields coming from the previous stage. Select the fields you want to combine. <ul style="list-style-type: none"> Note: The selected fields are displayed in the schema pane below the Search node field. • Search node: Type the name of the node to which you want to navigate in the schema pane. The typed node gets highlighted and its XML path is displayed in the XPath field below the pane. • Element: Click this button to convert an attribute to an element in the XML string. • Attribute: Click this button to convert an element to an attribute in the XML string. • Apply Namespace: If you want to specify an XML namespace for an element or an attribute, select it here. <ul style="list-style-type: none"> Note: You can create namespaces by clicking the Namespace button above. For details on this, see Defining Namespaces. • Remove: Click this button to remove the elements/attributes that you do not want in the XML string.
XPath	<p>Click anywhere in this field to view the XML path (XPath) of the node highlighted in the schema pane. To see all the previous XPaths viewed by you, click the down arrow at the right end of the field.</p> <p>Note: XPath is a language for finding information in an XML document. For further details on this, see https://www.w3schools.com/xml/xml_xpath.asp</p>

- **Runtime:** Use this button to specify multiple runtime instances of Field Combiner. This results in significant performance improvement.
- **OK:** Click this button to save all the details entered in this stage.
- **Cancel:** Click this button to cancel all the updates you made.
- **Help:** Click this button to refer to the help file for this stage.

Defining Namespace

Namespaces allow you to have duplicate element and attribute names in your output by assigning each element or attribute to an XML namespace.

To define one or more namespaces:

1. On the **Field Combiner Options** screen, click the **Namespace** button. The **Namespace Details** pop-up window is displayed.
2. In the **Prefix** column, enter the prefix to be associated with an element or attribute.
3. In the **Namespace** column, specify the URL of the namespace.
4. Repeat to define as many namespaces as you want to use.

Field Selector

Field Selector allows you to choose which fields to pass to the next stage in the dataflow. You can use Field Selector to remove unwanted fields from a dataflow. For example, if you have created a new field by combining the data from two fields, and you no longer need the two source fields, you can use Field Selector to retain only the new field and remove the two source fields from the dataflow.

Options

Option	Description
Select the fields to send to the next stage	Check the box next to each field that you want to send to the next stage in the dataflow. To prevent a field from being sent on to the next stage, clear the check box.
Select all	Check this box to select all the fields in the dataflow. Clear this box to deselect all fields.

Generate Time Dimension

Generate Time Dimension creates date records, one for each day of the date range you specify. You can then write these records to a time dimension table in a database using the Write to DB

stage. The time dimension table can then be used to perform accurate calculations based on a time period. For example, sales by quarter, budget spend by quarter, and revenue by day are all analyses that require a time dimension. Time dimension tables also enable you to account for fiscal years or non-standard quarters in the analysis.

Example Use of a Time Dimension Table

Time dimension tables are necessary for accurate time-based calculations because you sometimes cannot easily extract the necessary date data from the records. For example, the following records are in a sales database. Note that there are time gaps between records. For example, there is no record for the day 1/4/2012.

Date	Product	Amount
1/3/2012	Red Shirt	\$10.00
1/5/2012	Red Shirt	\$5.00
1/7/2012	Red Shirt	\$15.00

If you query these records and calculate the average sales per day, the answer would be \$10.00 ($\$30 / 3$ records). However, this is incorrect because the three records actually span a period of five days. If you have a time dimension table with a record for each day, you could join that table with the above table to get this:

Date	Product	Amount
1/3/2012	Red Shirt	\$10.00
1/4/2012		
1/5/2012	Red Shirt	\$5.00
1/6/2012		
1/7/2012	Red Shirt	\$15.00

Calculating the average sales per day using these records, you would get the correct answer: \$6.00 ($\$30 / 5$ days).

In addition, you could account for arbitrary time attributes such as holidays, weekends, and quarters in your calculation. For example, if 1/6/2012 happened to be a holiday and you were only interested in average sales per workday then the answer would be \$7.50.

Options

Generate Time Dimension has the following options.

Option	Description
Start date	The first day of the date range for which to generate time dimension records.
End date	Select this option if you want to specify a specific end date for the time dimension. One record will be generated for each day between the start date and the date you specify here.
Duration	Select this option if you want the time dimension to span a certain number of days, months, or years. One record will be generated for each day of the duration. For example, if you specify one week, seven records will be generated, starting with the day you specified in the Start date field.

Option	Description
--------	-------------

Time Attribute	
----------------	--

Option	Description
	Specifies the type of time information that you want to include in the time dimension. Each attribute will be a field in each day's record. One of the following:
Date	The day's date in the format <Date> <Month> <Year>, with the name of the month in the language of the server's locale setting. For example, if the server is running in an English locale, the date would read like this: 30 October 2012.
Day of Month	A number representing the day of the month. For example, 10 means the day is the 10 th day of the month.
Day of Year	A number representing the day of the year. For example, 304 means that the day is the 304 th day of the year.
Is Leap Year	A boolean value that indicates whether the day is in a leap year, in which case the value would be <code>true</code> , or not, in which case the value would be <code>false</code> .
Is Weekday	A boolean value that indicates whether the day is a weekday (Monday through Friday), in which case the value would be <code>true</code> , or not, in which case the value would be <code>false</code> .
Is Weekend	A boolean value that indicates whether the day is a weekend day (Saturday or Sunday), in which case the value would be <code>true</code> , or not, in which case the value would be <code>false</code> .
Julian Day	A unique number for the day. The Julian day system counts January 1, 4713 BC as day 1 and numbers all subsequent days sequentially since then. For example, 2456231 means that the day is the 2,456,231 st day since January 1, 4713 BC.
Julian Week	A unique number for the day's week. The Julian system counts the first week of 4713 BC as week 1 and numbers all subsequent weeks sequentially since then. For example, 350890 means that the day is in the 350,890 th week since the first week of 4713 BC.
Julian Year	A unique number for the year. The Julian system counts 4713 BC as year 1 and numbers all subsequent years sequentially since then. For example, 6725 means that the day is in the year 2012 AD.
Month Name	The name of the day's month in English.
Month Number	The number of the month of the year. For example, 3 means that the day is in the third month of the year.
Quarter	A number representing the quarter of the year. For example, 1 means that the day is in the first quarter of the year.
Week of Year	A number representing the week of the year. For example, 43 means that the day is in the 43 rd week of the year.
Weekday Name	The name of the day in English. For example, Monday.
Weekday Number	A number representing the day of the week, with day 1 being Monday. So for example Tuesday is day 2, Wednesday is day

Option	Description
	3, and so on.
	Year The day's year according to the Gregorian calendar. For example, 2012 means that the day is in the year 2012.
Field	The name of the field that will be created in the dataflow to contain the time attribute. A default field name is provided but you can modify the field name.
Type	The data type of this field. Generate Time Dimension automatically chooses the appropriate data type for each time attribute.
Calendar	Specifies whether to use a custom calendar when calculating the time attribute or the default Gregorian calendar. To define a custom calendar, click Calendar .

Creating a Calendar

A calendar defines important characteristics of the year in a way that is appropriate to the type of analysis you want to perform. By default, the calendar is the standard Gregorian calendar. However, the standard Gregorian calendar is not always appropriate. For example, if your company's financial year begins on June 1 instead of January 1, you could define the first month of the year as June, and the "month of year" time attribute would have June as month 1, July as month 2, and so on, as opposed to January as month 1, February as month 2, and so on.

- Do one of the following:
 - If you are configuring the Generate Time Dimension stage in Enterprise Designer, click **Calendars**.
 - In Management Console, go to **Resources > Calendars**.
- Click **Add**.
- In the **Calendar name** field, give the calendar a name that is meaningful to you.
- In the **Start month** field, select the first month of the year.
For example, if you select July, then July is month 1, August is month 2, and so on.

Note: Changing the start month affects how the values in the following fields are calculated: DayOfYear, MonthNumber, Quarter, and WeekOfYear.

- Click **Save**.

Output

Generate Time Dimension produces the following output fields.

Table 27: Generate Time Dimension Output

Field Name	Description
Date	The day's date in the format <Date> <Month> <Year>, with the name of the month in the language of the server's locale setting. For example, if the server is running in an English locale, the date would read like this: 30 October 2012.
DayOfMonth	A number representing the day of the month. For example, 10 means the day is the 10 th day of the month.
DayOfYear	A number representing the day of the year. For example, 304 means that the day is the 304 th day of the year.
IsLeapYear	A boolean value that indicates whether the day is in a leap year, in which case the value would be <code>true</code> , or not, in which case the value would be <code>false</code> .
IsWeekday	A boolean value that indicates whether the day is a weekday (Monday through Friday), in which case the value would be <code>true</code> , or not, in which case the value would be <code>false</code> .
IsWeekend	A boolean value that indicates whether the day is a weekend day (Saturday or Sunday), in which case the value would be <code>true</code> , or not, in which case the value would be <code>false</code> .
JulianDay	A unique number for the day. The Julian day system counts January 1, 4713 BC as day 1 and numbers all subsequent days sequentially since then. For example, 2456231 means that the day is the 2,456,231 st day since January 1, 4713 BC.
JulianWeek	A unique number for the day's week. The Julian system counts the first week of 4713 BC as week 1 and numbers all subsequent weeks sequentially since then. For example, 350890 means that the day is in the 350,890 th week since the first week of 4713 BC.
JulianYear	A unique number for the year. The Julian system counts 4713 BC as year 1 and numbers all subsequent years sequentially since then. For example, 6725 means that the day is in the year 2012 AD.

Field Name	Description
MonthName	The name of the day's month in English.
MonthNumber	The number of the month of the year. For example, 3 means that the day is in the third month of the year.
Quarter	A number representing the quarter of the year. For example, 1 means that the day is in the first quarter of the year.
WeekOfYear	A number representing the week of the year. For example, 43 means that the day is in the 43 rd week of the year.
WeekdayName	The name of the day in English. For example, Monday.
WeekdayNumber	A number representing the day of the week, with day 1 being Monday. So for example Tuesday is day 2, Wednesday is day 3, and so on.
Year	The day's year according to the Gregorian calendar. For example, 2012 means that the day is in the year 2012.

Query Cache

Query Cache looks up data in a cache based on values in one or more dataflow fields and returns data from matching records in the cache, adding the cache record's data to the record in the dataflow. Looking up data in a cache can improve performance compared to looking up data in a database.

There are two kinds of caches: global caches and local caches.

Global Cache Options

A global cache is system-wide, shared cache that will reside in memory. Choose a global cache if you want the cache to be available to multiple dataflows or when data does not change often or remains relatively static and when storage is not limited. A global cache is static as you can write to it only once. The cache can not be updated once it has been created.

A global cache is created by the Write to Cache stage. Before you can use a global cache you must populate the cache with the data you want to look up. To do this, create a dataflow containing the **Write to Cache** stage.

Option Name	Description
Cache type	Select the Global cache option.
Cache name	Specifies the cache you want to query. To create a cache, use the Write to Cache stage.
Cache Fields	This column lists the fields in the cache. You cannot modify these field names.
Stage Fields	This column lists the field names used in the dataflow. If you wish to change a field name, click the field name and enter a new name.
Type	This column lists the data type of each dataflow field.
Include	Check the box in this column to have the query return the value of the cache field. Clear the box if you do not want the query to return the cache field.

Option Name Description

Default Error Value Specifies the value to be displayed in the dataflow field if the query fails. The drop-down list displays valid values corresponding to data type of the queried field. For example, in case of an **integer** the option displayed is **-1**.

You can also enter a value to this field. See the table below for a list of valid default error values for various data types.

Data type Valid Default Error Value along with data type (in bracket)

Data type	Valid Default Error Value along with data type (in bracket)
Null	-1 (Integer)
	1899-12-30 12:00:00 (Date Time)
	1899-12-30 12:00:00 (Date Time)
	False
	Empty

Data type	Valid Default Error Value along with data type (in bracket)
Date	
Integer	✓
Long	✓
Float	✓
Big Decimal	✓
Double	✓
String	✓ ✓ ✓ ✓ ✓ ✓ ✓
Time	✓
Date Time	✓
Boolean	✓

Option Name	Description
Key Field	Specifies the field in the cache that will be used as a lookup key. If the value in the field in the Input Field column matches the value in the key field in the cache, then the query returns data from that record in the cache.
Input Field	Specifies the dataflow field, the value of which will be used as a key. If the value in this field matches the value in the key field in the cache, then the query returns data from that record in the cache.

Local Cache Options

A local cache is a temporary cache which is only used during the execution of Query Cache stage. The Query Cache builds the cache from the database table you choose. It then looks up data in the cache based on key fields and lookup conditions and returns data from matching records in the cache, adding the cache record's data to the record in the dataflow.

A local cache is dynamic as it is created during a job execution of the Query Cache. Once Query Cache completes reading the data, the cache is automatically deleted from the memory. A local cache is recreated every time the Query Cache stage is executed. Choose a local cache if it is only going to be used in one dataflow or if the lookup table changes frequently.

Option name	Description						
Cache type	Specifies the Local cache option.						
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Spectrum Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage Connections.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <table border="0"> <tr> <td>Connection name</td> <td>Enter a name for the connection. The name can be anything you choose.</td> </tr> <tr> <td>Database driver</td> <td>Select the appropriate database type.</td> </tr> <tr> <td>Connection options</td> <td>Specify the host, port, instance, user name, and password to use to connect to the database.</td> </tr> </table>	Connection name	Enter a name for the connection. The name can be anything you choose.	Database driver	Select the appropriate database type.	Connection options	Specify the host, port, instance, user name, and password to use to connect to the database.
Connection name	Enter a name for the connection. The name can be anything you choose.						
Database driver	Select the appropriate database type.						
Connection options	Specify the host, port, instance, user name, and password to use to connect to the database.						
Table/view	Specifies the table or view in the database that you want to query.						
Database Fields	This column lists the fields in the database. You cannot modify these field names.						

Option name	Description
Stage Fields	This column lists the field names used in the dataflow. If you wish to change a field name, click the field name and enter the new name.
Type	This column lists the data type of each dataflow field.
Include	Check the box in this column to have the query return the value of the cache field. Clear the box if you do not want the query to return the cache field.

Option name Description

Default Error Value Specifies the value to be displayed in the dataflow field if the query fails. The drop-down list displays valid values corresponding to data type of the queried field. For example, in case of an **integer** the option displayed is **-1**.

You can also enter a value to this field. See the table below for a list of valid default error values for various data types.

Data Valid Default Error Value along with data type (in bracket)
type

Data type	Null	-1	1899-12-30	1899-12-30	12:00:00	False	Empty
Date							
Integer		✓					
Long		✓					
Float		✓					
Big Decimal		✓					
Double		✓					
String	✓	✓	✓	✓	✓	✓	✓
Time					✓		
Date Time			✓				
Boolean						✓	

Option name	Description
Key Field	Specifies the field in the database that will be used as a look up key. If the value in the field in Input field column matches the value in the Key field in the database, then the query returns the data from that record in the database.
Type	Data type of the Key Field value
Operator	Select the required operator. The supported operators are: <ul style="list-style-type: none"> • = • != • > • >= • < • <=
Is Constant	Select this check box if you want the query to return value based on a constant you enter, instead of the Input Field.
Input Field	Specifies the dataflow field whose value will be used as a key. If the value in this field matches the value in the Key field in the database, then the query returns data from that record in the database..

Advanced Cache Options

An advanced cache is a temporary cache similar to local cache. It is used during the execution of Query Cache stage. It builds the cache based on the SQL Query which reads the data from the tables mentioned in the query. It then looks up data in the cache based on the lookup keys mentioned in the where clause and returns data from matching records in the cache, adding the cache record's data to the record in the dataflow..

An advanced cache is dynamic as it is created during a job execution of the Query Cache. Once Query Cache completes reading the data, the cache is automatically deleted from the memory. An advanced cache is recreated every time the Query Cache is executed. Choose an advanced cache option if it is going to read the data from multiple tables and there is some complex query needs to be executed for cache creation.

Option name	Description
Cache type	Specifies the Advanced cache option.

Option name	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection Name Enter a name for the connection. The name can be anything you choose.</p> <p>Database Driver Select the appropriate database type.</p> <p>Connection Options Specify the host, port, instance, user name, and password to use to connect to the database.</p>
Query	<p>Provides SQL query to read data from the database. The query can read data from multiple tables.</p> <p>Note: Providing alias is mandatory in the query.</p>
Where	<p>This text is used as the where clause to lookup the cache created based on Query. User can specify input field in the Query using \$ operator as prefix. For example, <code>_id = \${_inputId}</code>, Where <code>_inputId</code> is the input field and <code>_id</code> is the lookup column in the cache.</p>
Get Fields	<p>This populates the grid with the fields which are selected to be cached using SQL query.</p>
Database Fields	<p>This column lists the fields fetched in the database. You cannot modify these field names.</p>
Stage Fields	<p>This column lists the field names used in the dataflow. If you wish to change a field name, click the field name and enter the new name.</p>
Type	<p>This column lists the data type of each dataflow field.</p>

Option name Description

Default Error Value Specifies the value to be displayed in the dataflow field if the query fails. The drop-down list displays valid values corresponding to data type of the queried field. For example, in case of an **integer** the option displayed is **-1**.

You can also enter a value to this field. See the table below for a list of valid default error values for various data types.

Data Valid Default Error Value along with data type (in bracket)
type

Data type	Null	-1	1899-12-30	1899-12-30	12:00:00	False	Empty
Date							
Integer		✓					
Long		✓					
Float		✓					
Big Decimal		✓					
Double		✓					
String	✓	✓	✓	✓	✓	✓	✓
Time					✓		
Date Time			✓				
Boolean						✓	

Runtime Tab

The options available in Runtime tab are common for global, local, and advanced caches.

Option Name	Description
Match options	<p>Specifies what to do if there is more than one record in the cache that matches the query.</p> <p>Return all matches Return data from all records in the cache that have a matching value in the key field or fields.</p> <p>Return the first matching record Return data from only the first record in the cache that has a matching value in the key field or fields.</p> <p>Return the last matching record Return data from only the last record in the cache that has a matching value in the key field or fields.</p>
Stage Options	<p>This section lists the dataflow options used in the SQL query of this stage and allows you to provide a default value for all these options. The Name column lists the options while you can enter the default values in the corresponding Value column.</p> <p>Note: The default value provided here is also displayed in the Map dataflow options to stages section of the Dataflow Options dialog box. The dialogue box also allows you to change the default value. In case of a clash of default values provided for an option through Stage Options, Dataflow Options, and Job Executor the order of precedence is: Value provided through Job Executor > Value defined through the Dataflow Options dialogue box > Value entered through the Stage Options.</p>

Query DB

The Query DB stage allows you to use fields as parameters into a database query and return the results of the query as new fields in the dataflow.

Note: If you want to query a spatial database, use Query Spatial Data instead of Query DB.

General Tab

Option	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Spectrum Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage Connections.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection name Enter a name for the connection. The name can be anything you choose.</p> <p>Database driver Select the appropriate database type.</p> <p>Connection options Specify the host, port, instance, user name, and password to use to connect to the database.</p>
Table/View	Specifies the table or view in the database that you want to query.
Where	<p>If you want to use a <code>WHERE</code> statement, enter it here. Note that you should not actually include the word <code>WHERE</code> in the statement. The purpose of a <code>WHERE</code> statement is to return only the data from records that match the condition you specify.</p> <p>To specify a value from a dataflow field, use this syntax:</p> <pre> <code>\${field name}</code> </pre> <p>Where <code>field name</code> is the name of a field in the dataflow.</p> <p>For example:</p> <pre> <code>account_number=\${customer_key}</code> </pre> <p>In this example, the query would return data from records where the value in the table column <code>account_number</code> matches the value in the dataflow field <code>customer_key</code>.</p> <p>Note: If you are querying a case-sensitive database, make sure you enter the field name in the same format as used in the database table. In other words, enclose the field name in quotes (") if the field names were quoted during table creation.</p> <p>Click Preview to see a preview of the data (first 50 records) based on the criteria you defined.</p> <p>Note: The preview feature in Query DB does not work if you use a dataflow field in the <code>WHERE</code> statement. Instead, you can preview the result using the dataflow inspection tool in Spectrum Enterprise Designer.</p>

Option	Description
Return records with no results	Check this box if you want records whose queries return no results to still be returned by Query DB. If you clear this check box, the record will not be returned. We recommend that you leave this option checked.
Include	In the fields table, select the fields you want to include by clicking the Include box next to the field.

Sort Tab

If you want to sort records based on the value of a field, specify the fields you want to sort on.

Parameterizing Query DB at Runtime

You can configure the Query DB stage so that values in the `WHERE` clause are specified at runtime. This is useful in cases where you want to make the column name in the `WHERE` clause configurable using Dataflow Options.

1. Open the dataflow in Spectrum Enterprise Designer.
2. Configure the **Connection** and **Table/View** name fields to point to the database you want to query.
3. In the **Where** field, enter a `WHERE` statement using the following format for values you want to parameterize: `${parameter}`.

For example:

```
${COL}=${EmployeeID}
```

Here `COL` represents a Dataflow option which will be populated with the column name for the table at runtime.

4. Close the Query DB options window.
5. Click the Dataflow Options icon on the toolbar or click **Edit > Dataflow Options**. The **Dataflow Options** window appears.
6. Click **Add**. The **Define Dataflow Options** window appears.
7. Select the Query DB stage.
8. Specify **Option name** and **Option label**.

The value in the **Option name** field should be the same value as entered in the format `${parameter}` in the `WHERE` clause. In the **Option label** field, you can specify a different label or keep it the same as the **Option name**.

For example: `COL`

9. Specify the **Default value**. For example: "EmpID".
10. Click **OK**.

This procedure maps the actual database column name i.e. EmpID, to the runtime option name "COL". The database column name needs to be appropriately quoted with the specific quote identifier for the particular database.

Query NoSQL DB

The **Query NoSQL DB** stage allows lookups of required data from a NoSQL database. The stage supports MongoDB databases.

General Tab

Field Name	Description
Connection	<p>Select the required database connection from the dropdown list. The options displayed are based on the connections defined in Spectrum Management Console.</p> <p>To add a new connection, see Connecting to NoSQL.</p> <p>To modify an existing connection, select and open it from the list of connections on the Connections page of Spectrum Management Console, make the required updates, and click the Save button.</p>
Table/View	<p>Specifies the collection or view in the database that you want to query.</p> <p>Note: In a MongoDB database, a table/view is called a <i>collection</i>.</p>
Schema File	<p>Click the Browse button (...) to select a JSON Schema file. This file is optional. Fields in the fields tab can be regenerated either using the schema file or database table/view.</p> <p>To clear the selected file path, click Clear.</p> <p>Note: Fields will always be generated using the schema file if one is selected.</p>
Where	<p>Enter the <code>where</code> clause to define the criteria for the lookup.</p> <p>For a list of the supported operators in the <code>WHERE</code> clause, see http://docs.mongodb.org/manual/reference/operator/query/.</p>

Field Name	Description
Preview	Displays the records from the selected table or view. Note: Clicking Preview fetches the first 50 records from the selected database, without applying the filter criteria specified in the Where field.
Expand All	Expands the items in the preview tree.
Collapse All	Collapses the items in the preview tree.

Fields Tab

The Fields tab allows you to select the data that you want to pass to the next stage. For more information, see [Defining Fields - Query NoSQL DB](#) on page 297.

Defining Fields - Query NoSQL DB

The **Fields** tab displays the field and its type as defined in the NoSQL DB/Schema file.

1. On the **Fields Tab**, click **Regenerate**.

This generates the aggregated data based on the first 50 records. The data is displayed in the following format: `Fieldname (datatype)`.

Note: If the schema file is browsed, the fields are generated using the schema file. The table/view is bypassed. To reset the schema file, click **Clear**.

2. To modify the name and type of a field, highlight the field, and click **Modify**.
3. In the **Name** field, choose the field you want to add or type the name of the field.
4. In the **Type** field, you can leave the data type as string if you do not intend to perform any mathematical operations with the data. However, if you intend to perform these kinds of operations, select an appropriate data type. This will convert the string data from the file to a data type that will enable the proper manipulation of the data in the dataflow.

The stage supports the following data types:

double A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. 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 $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -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.

- You can also add fields that are not present in the table or schema file. Click **Add** to add a new field. To remove a field, click **Remove**.

Note: You can only add a new field under the List type.

- Click **OK**.

Configuring Dataflow Options - Query NoSQL DB

This procedure describes how to configure a dataflow to support runtime options for **Query NoSQL DB**.

- Open the flow in Spectrum Enterprise Designer.
- If you want to configure runtime options for a stage in an embedded flow, open the embedded flow.
- Click the Dataflow Options icon on the toolbar or click **Edit > Dataflow Options**. The **Dataflow Options** dialog box appears.
- Click **Add**. The **Define Dataflow Options** dialog box appears.
- Expand the **Query NoSQL DB** stage.
- The below dataflow options are exposed to query a Mongo DB database:

- Connection
- Table

The selected **Query NoSQL DB** option name is displayed in **Option name** and **Option label** fields. This is the option name that will have to be specified at runtime in order to set this option.

- Enter a description of the option in the **Description** field.
- In the **Target** field, select the option **Selected stage(s)**.
- 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.
- 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 845.

- Click **OK**.
- Continue adding options as desired.

13. Click **OK** in the **Dataflow Options** dialog box when you are done adding options.
14. Save and expose the dataflow.

Read From DB

The **Read From DB** stage reads data from a database table or view as input to a dataflow. The stage is available for jobs, services, and subflows but not for process flows.

Note: The stage supports reading data from and writing data to HDFS 3.x and Hive 2.1.1. The support includes:

- Connectivity to Hive from Spectrum on Windows
- Support and connectivity to Hive version 2.1.1 from Spectrum with high availability
- Support to Read and Write from Hive DB (JDBC) via Model Store connection

Also see [Best Practices for connecting to HDFS 3.x and Hive 2.1.1](#).

Related task:

To be able to use the **Read from DB** stage, you need to create connection to the respective database using the **Connection Manager** of **Spectrum Management Console**. For details, see Database Connection Manager in [Write to DB](#) on page 393.

General Tab

Field Name	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Spectrum Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage Connections.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection name Enter a name for the connection. The name can be anything you choose.</p> <p>Database driver Select the appropriate database type.</p> <p>Connection options Specify the host, port, instance, user name, and password to use to connect to the database.</p>

Field Name	Description
SQL	<p>Enter the SQL query to specify the records that need to be read from the data source on running the dataflow. You can manually type the SQL query in this field. Alternatively, use the Visual Query Builder to construct the query by clicking Build SQL.</p> <p>The SQL query can include variables instead of actual column names. Using variables allows you to customize the query at runtime. For more information, see Query Variables on page 304.</p> <p>A sample query for exposing <i>BranchID</i> and <i>BranchType</i> from <i>public.Branch</i> table in the database can be:</p> <pre>select * from "public"."Branch" where "BranchID" = # {ID} and "BranchType" = # {Type}</pre> <p>Note: For <i>Integer</i> type fields values can be entered without quotes but for <i>String</i> type it should be in single quotes.</p>
Build SQL	<p>Create a complex query by selecting multiple columns, and creating joins and nested queries by clicking Build SQL. The Visual Query Builder opens. For more information, see Visual Query Builder on page 301.</p> <p>Note: A query created using the Visual Query Builder is displayed with fully qualified names of columns and tables in the SQL field.</p>
Regenerate Fields	<p>To see the schema of the data to be fetched by the query, click Regenerate Fields. If you edit an existing query, click Regenerate Fields to fetch the modified schema.</p> <p>Note: On clicking Regenerate Fields, the entity names in the SQL query are retained and not replaced with their fully qualified names.</p>
Preview	To see a sample of the records fetched by the SQL query, click Preview .

Note: The **Read From DB** stage allows you to modify the type of an input field.

Note: The **Read from DB** stage reads all values of the `date` datatype as `String` values. This is the behavior of the *JTDS driver*, which is the default driver used by Spectrum. To handle all `date` datatype values as is, use Microsoft's JDBC driver.

Runtime Tab

Field name	Description
Fetch size	<p>Select this option to specify the number of records to read from the database table at a time. For example, if the Fetch size value is 100 and total number of records to be read is 1000, then it would take 10 trips to the database to read all the records.</p> <p>Setting an optimum fetch size can improve performance significantly.</p> <p>Note: You can calculate an optimum fetch size for your environment by testing the execution times between a Read from DB stage and a Write to Null stage. For more information, see Determining an Optimum Fetch Size.</p>
Stage Options	<p>This section lists the dataflow options used in the SQL query of this stage and allows you to provide a default value for all these options. The Name column lists the options while you can enter the default values in the corresponding Value column.</p> <p>Note: The default value provided here is also displayed in the Map dataflow options to stages section of the Dataflow Options dialog box. The dialogue box also allows you to change the default value. In case of a clash of default values provided for an option through Stage Options, Dataflow Options, and Job Executor the order of precedence is: Value provided through Job Executor > Value defined through the Dataflow Options dialogue box > Value entered through the Stage Options.</p>

Visual Query Builder

Visual Query Builder provides a visual interface for building complex SQL queries in the Read from DB stage. To work with Visual Query Builder, you need basic knowledge of SQL concepts.

To access Visual Query Builder, click the **Build SQL** button in Read from DB.

The query builder main window is divided into these parts:

- The **Query Building Area** is the main area where the visual representation of query will be displayed. This area allows you to define source database objects, define links between them and configure properties of tables and links.
- The **Columns Pane** is located below the query building area. It is used to perform all necessary operations with query output columns and expressions. Here you can define field aliases, sorting and grouping, and define criteria.
- The page control above the query building area will allow you to switch between the main query and sub-queries.

Adding Objects to a Query

To add an object to a query, use the **Database Objects** tree. The objects within the tree are grouped by database, schema, and type. Drag the object you want to add to the query and drop it to the query building area. Alternatively, you can also double click the object to add it to the query.

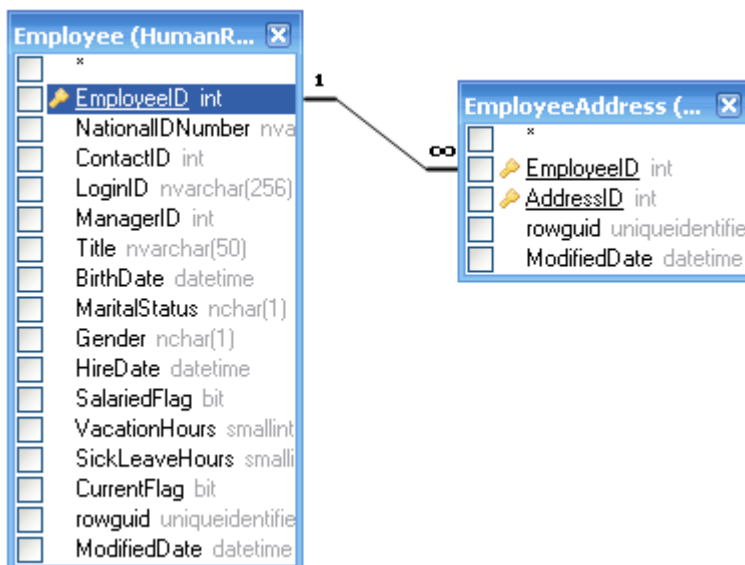
Setting Object Aliases

To set an alias for an object or derived table in the query, double-click the object and select **Properties**. The **Datasource Properties** dialog appears. It may contain other server-specific datasource options, but the Alias property is the same for all database servers.

Joining Tables

When two objects referenced with a foreign key relationship are added to the query, they become joined automatically with INNER JOIN. For those servers that do not support the JOIN clause, the query builder adds a condition to the WHERE part of the query.

To join two objects manually, select the field by which you want to link the object with another and drag it to the corresponding field of the other object. After you finish dragging, a line connecting the linked fields will appear. Key cardinality symbols are placed at the ends of link when a corresponding relationship exists in the database.



To remove a link between objects, double-click the link line and select **Remove**.

To change the join type, double click the link line.

Selecting Output Fields

To add a field to the list of query output fields, check the box at the left of the field name in the datasource field list in the **Query Building** area. To include all the fields of the object, check the box

at the left of the asterisk item in the datasource field list. You may also drag fields from the **Query Building** area to the **Columns** pane to get the same result.

If you do not select any fields from the query datasources, an asterisk item will be added to the select list of the resulting query ("Select * From ..."). This is because a SELECT query without any columns will produce an error for most database servers. The asterisk item is removed from the query if you select any field or add any output expression to the query.

Tip: Another way to add a field is to select a field name from the drop-down list of the Expression column in the **Columns** pane. You may also type any valid expression in the **Expression** column in the **Columns** pane. To insert an empty line to the **Columns** pane, press the Alt+Insert key.

To remove a field from the **Columns** pane, clear the check box at the left of the field name in the **Query Building** area or press the Alt+Delete key in the **Columns** pane.

To move a line up press the Alt+Up key. To move a line down press the Alt+Down key.

To remove an expression from the SELECT list of the query, clear the check box in the **Output** column.

To set an alias for an expression, enter the alias in the **Alias** column. Aliases become the headings of columns in the resulting dataset.

Sorting a Dataset

To sort the resulting dataset, use the Sort Type and Sort Order columns of the **Columns** pane. The Sort Type column allows you to sort in ascending or descending order. The Sort Order column allows you to set up the order in which fields will be sorted, if more than one field will be sorted.

To disable sorting by a field, clear the Sort Type column for the field.

Defining Criteria

To define criteria, use the **Criteria** column and the **Or** columns of the **Columns** pane. When writing conditions in these columns, omit the expression itself. For example, to get the following criteria in your query:

```
WHERE (Field1 >= 10) AND (Field1 <= 20)
```

Type the following in the Criteria cell of the Field1 expression:

```
>= 10 AND <= 20
```

Criteria placed in the **Or** columns will be grouped by columns using the AND operator and then concatenated in the WHERE (or HAVING) clause using the OR operator. For example, the criteria shown below will produce the SQL statement below. Please note that criteria for Field1 is placed both to the Criteria and Or columns.

Output	Expression	Aggregate	Alias	Sort Type	Sort Order	Grouping	Criteria	Or...	Or...
<input checked="" type="checkbox"/>	Field1					<input type="checkbox"/>	= 10	= 10	
<input checked="" type="checkbox"/>	Field2					<input type="checkbox"/>	< 0	> 10	
<input type="checkbox"/>						<input type="checkbox"/>			

```
WHERE (Field1= 10) AND ((Field2 < 0) OR (Field2 > 10))
```

Some expressions may be of Boolean type, for example the EXISTS clause. In this case you should type "= True" in the Criteria column of such expressions or "= False" if you want to place a NOT operator before the expression.

Grouping Output Fields

To build a query with grouping, mark expressions for grouping with the **Grouping** check box.

A query with grouping may have only grouping or aggregate expressions in the SELECT list. Thus the query builder allows you to set the Output check box for grouping and aggregate expressions. If you try to set this check box for a column without the grouping or aggregate function set, a Grouping check box will be set automatically to maintain the validity of resulting SQL query.

When the **Columns** pane contains columns marked with the **Grouping** check box, a new column called **Criteria for** appears in the grid. This column applies criteria to expression groups or to their values.

For example, you have a column "Quantity" with Aggregate function "Avg" in your query and you type > 10 in the Criteria column. Having the "for groups" value set in the Criteria for column, the resulting query will contain only groups with an average quantity greater than 10, and your query will have the "Avg(Quantity) > 10" condition in a HAVING clause. Having the "for values" value set in the Criteria for column, the result query will calculate the Average aggregate function only for records with Quantity value greater than 10, and your query will have the "Quantity > 10" condition in WHERE clause.

Defining SQL Query Properties

You can define options specific to your database server by using the context popup menu of the **Query Building** area.

Query Variables

In the **Read From DB** stage, while defining the query, you can include variables instead of actual column names. Using variables in the query allows you to customize the query conditions at runtime (using the **Dataflow Options**) or through the **Job Executor**.

However, you can also provide Stage Options value in the **Runtime** tab and view the schema and sample of records to be fetched by the query by using the **Regenerate Fields** and **Preview** buttons respectively.

A variable is defined using the format `#{variable}`, and inserted in either the `select` or `where` clause of an SQL query.

Note: You can edit a query generated using the **Visual Query Builder** to include variables. However, the edited query will not be readable by the Visual Query Builder anymore. The **Build SQL** button is disabled when you include a variable in a manually written or generated SQL query.

Inserting a Query Variable

1. Open the required job, which includes a **Read From DB** stage. Alternatively, add a **Read From DB** stage to the job.
2. Open the **Read From DB Options** of the **Read From DB** stage.
3. Create the SQL query in the **SQL** field, either manually or using the Visual Query Builder. For more information, see **Visual Query Builder** on page 301.
4. Add the desired conditions in the `where` clause of the query using variables with the syntax `#{variable}`.

For example, in a table `CUSTOMERS`, which has the column `AGE` with values such as 28, 32, 30, and so forth, and a column `SALARY` with values such as 1000, 1500, 2200, and so on, frame an SQL query as below:

```
select * from CUSTOMERS where #{condition1} > 28 and #{condition2} > 1200
```

Note: On inserting a variable in the `where` clause of the SQL query, the **Build SQL...** button is disabled.

5. To view the schema and the sample records to be fetched by the query, enter Stage Options value on the **Runtime** tab, and then click the **Regenerate Fields** and **Preview** buttons respectively.
6. Click **OK**.

The `where` clause of the SQL query can now be customized at runtime using the **Dataflow Options**, or while executing the job through the JobExecutor.

Note: A variable can be placed in the `select` clause of an SQL query as well. However, such a variable name should match the name of one of the columns of the table being queried.

Configuring a Query Variable as a Dataflow Option

1. Open the required job for which the query containing the variable(s) has been defined in a **Read From DB** stage.
2. Open **Edit > Dataflow Options....**
3. Click **Add**.
4. In the **Map dataflow options to stages** section, expand the **Read From DB** entry. The variables defined in the SQL query in the **Read From DB** stage are listed along with the other attributes of the stage.
5. Select the variable you wish to customize using the corresponding checkbox.

6. Enter a relevant name for the variable in the **Option label** field.
7. In the **Default value** field, enter the column name which is to be used instead of the selected variable in the `where` clause of the SQL query. Alternatively, enter a constant value to be used instead of the variable in the `where` clause.

For example, for the below SQL query defined in the **Read From DB** stage:

```
select * from CUSTOMERS where #{condition1} > 28 and #{condition2} > 1200
```

you can select the column `AGE` of the table `CUSTOMERS` as the **Default value** for the variable `condition1`, and the column `SALARY` as the **Default value** for the variable `condition2`.

At runtime, the query is interpreted as:

```
select * from CUSTOMERS where AGE > 28 and SALARY > 1200
```

8. Repeat steps 5-7 for all the variables placed in the SQL query in the **Read From DB** stage.
9. Click **OK**.

On running the dataflow, the customized query is used to fetch the required data.

Configuring a Query Variable for Job Executor

Note: Ensure you have the Spectrum Job Executor downloaded to your server.

1. Create a text file that defines the default values of the variables included in the SQL query of the **Read From DB** stage of the job.

To assign a column name `AGE` as the default value to a variable `condition1`, create a text file, say `variables.txt`, and include the below line in the file:

```
condition1=AGE
```

To assign a constant value, say `20`, as the default value to a variable `condition1`, include the below line in the file:

```
condition1=20
```

2. While running the job using the Job Executor on the command prompt, use the argument `-o` followed by the path to the created text file.
For example, to run the job `ReadCustomerDataJob`, for which the default values of its variables have been defined in the text file `variables.txt`, run the below command on a command prompt:

```
java -jar jobexecutor.jar -h "localhost" -u "admin" -p "admin" -s "8080" -j "ReadCustomerDataJob" -o "variables.txt"
```

On running the job using the Job Executor, the customized query is used to fetch the required data.

Note: For instructions and command-line syntax, see *Running a Job* in the Spectrum Dataflow Designer Guide.

Read From File

The **Read from File** stage specifies an input file for a job or subflow. It is not available for services.

Note: If you want to use an XML file as input for your dataflow, use the Read from XML stage instead of Read from File. If you want to use a variable format file as input, use Read from Variable Format File.

Prerequisite: To read a file from any of the file system connection types, such as FTP, Cloud, Amazon AWS S3, and HDFS, perform these steps:

1. Create a connection to these file servers using **Spectrum Management Console** or **Discovery**. For details, see section **Defining Connections**.
2. Select the file using the **File name** field in **File Properties** tab (described below).

File Properties Tab

Field Name	Description
Server name	Indicates whether the file you select as input is located on the computer running Spectrum Enterprise Designer or on the Spectrum Technology Platform server. If you select a file on the local computer, the server name will be My Computer. If you select a file on the server the server name will be Spectrum Technology Platform.
File name	<p>Specifies the path to the file. Click the ellipses button (...) to go to the file you want.</p> <p>You can read multiple files by using a wild card character to read data from multiple files in the directory. The wild card characters * and ? are supported. For example, you could specify *.csv to read in all files with a .csv extension located in the directory. In order to successfully read multiple files, each file must have the same layout (the same fields in the same positions). Any record that does not match the layout specified on the Fields tab will be treated as a malformed record.</p> <p>While reading a file from an HDFS file server, the compression formats supported are:</p> <ol style="list-style-type: none"> 1. GZIP (.gz) 2. BZIP2 (.bz2) <p>Note: The extension of the file indicates the compression format to be used to decompress the file.</p> <p>Attention: If the Spectrum Technology Platform server is running on Linux, remember that file names and paths on these platforms are case sensitive.</p>

Field Name	Description
Record type	<p>The format of the records in the file. Select one of:</p> <p>Line Sequential A text file in which records are separated by an end-of-line (EOL) character such as a carriage return or line feed (CR or LF) and each field has a fixed starting and ending character position.</p> <p>Fixed Width A text file in which each record is a specific number of characters in length and each field has a fixed starting and ending character position.</p> <p>Delimited A text file in which records are separated by an end-of-line (EOL) character such as a carriage return or line feed (CR or LF) and each field is separated by a designated character such as a comma.</p>

Field Name	Description
Character encoding	The text file's encoding. Select one of these:
CP1252	This encoding is also known as the Windows-1252 or simply Windows character set. It is a super set of ISO-8859-1 and uses the 128-159 code range to display additional characters not included in the ISO-8859-1 character set.
UTF-8	Supports all Unicode characters and is backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html .
UTF-16	Supports all Unicode characters but is not backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html .
US-ASCII	A character encoding based on the order of the English alphabet.
UTF-16BE	UTF-16 encoding with big endian byte serialization (most significant byte first).
UTF-16LE	UTF-16 encoding with little endian byte serialization (least significant byte first).
ISO-8859-1	An ASCII character encoding typically used for Western European languages. Also known as Latin-1.
ISO-8859-3	An ASCII character encoding typically used for Southern European languages. Also known as Latin-3.
ISO-8859-9	An ASCII character encoding typically used for Turkish language. Also known as Latin-5.
CP850	An ASCII code page used to write Western European languages.
CP500	An EBCDIC code page used to write Western European languages.
Shift_JIS	A character encoding for the Japanese language.
MS932	A Microsoft's extension of Shift_JIS to include NEC special characters, NEC selection of IBM extensions, and IBM extensions.
CP1047	An EBCDIC code page with the full Latin-1 character set.

Field Name	Description						
Field separator	<p>Specifies the character used to separate fields in a delimited file. For example, this record uses a pipe () as a field separator:</p> <pre>7200 13TH ST MIAMI FL 33144</pre> <p>These characters available to define as field separators are:</p> <ul style="list-style-type: none"> • Space • Tab • Comma • Period • Semicolon • Pipe <p>If the file uses a different character as a field separator, click the ellipses button to select another character as a delimiter.</p>						
Text qualifier	<p>The character used to surround text values in a delimited file. For example, this record uses double quotes (") as a text qualifier.</p> <pre>"7200 13TH ST" "MIAMI" "FL" "33144"</pre> <p>The characters available to define as text qualifiers are:</p> <ul style="list-style-type: none"> • Single quote (') • Double quote (") <p>If the file uses a different text qualifier, click the ellipses button to select another character as a text qualifier.</p>						
Record separator	<p>Specifies the character used to separate records in line a sequential or delimited file. This field is not available if you check the Use default EOL check box.</p> <p>The record separator settings available are:</p> <table border="0"> <tr> <td>Linux (U+000A)</td> <td>A line feed character separates the records. This is the standard record separator for Linux systems.</td> </tr> <tr> <td>Macintosh (U+000D)</td> <td>A carriage return character separates the records. This is the standard record separator for Macintosh systems.</td> </tr> <tr> <td>Windows (U+000D U+000A)</td> <td>A carriage return followed by a line feed separates the records. This is the standard record separator for Windows systems.</td> </tr> </table> <p>If your file uses a different record separator, click the ellipses button to select another character as a record separator.</p>	Linux (U+000A)	A line feed character separates the records. This is the standard record separator for Linux systems.	Macintosh (U+000D)	A carriage return character separates the records. This is the standard record separator for Macintosh systems.	Windows (U+000D U+000A)	A carriage return followed by a line feed separates the records. This is the standard record separator for Windows systems.
Linux (U+000A)	A line feed character separates the records. This is the standard record separator for Linux systems.						
Macintosh (U+000D)	A carriage return character separates the records. This is the standard record separator for Macintosh systems.						
Windows (U+000D U+000A)	A carriage return followed by a line feed separates the records. This is the standard record separator for Windows systems.						

Field Name	Description
Use default EOL	<p>Specifies that the file's record separator is the default end of line (EOL) character used on the operating system on which the Spectrum Technology Platform server is running.</p> <p>Do not select this option if the file uses an EOL character that is different from the default EOL character used on the server's operating system. For example, if the file uses a Windows EOL but the server is running on Linux, do not check this option. Instead, select the Windows option in the Record separator field.</p>
Record length	<p>For fixed width files, specifies the exact number of characters in each record.</p> <p>For line sequential files, specifies the length, in characters, of the longest record in the file.</p>
First row is header record	<p>Specifies whether the first record in a delimited file contains header information and not data.</p> <p>For example, this file snippet shows a header row in the first record.</p> <pre>"AddressLine1" "City" "StateProvince" "PostalCode" "7200 13TH ST" "MIAMI" "FL" "33144" "One Global View" "Troy" "NY" 12180</pre>
Treat records with fewer fields than defined as malformed	<p>Delimited file records containing fewer fields than are defined on the Fields tab will be treated as malformed.</p>
Import	<p>Imports the file layout definition, encoding setting, and sort options from a settings file. The settings file is created by exporting settings from another Read from File or Write to File stage that used the same input file or a file that has the same layout as the file you are working with.</p>
Export	<p>Saves the file layout definition, encoding setting, and sort options to a settings file. You can then import these settings into other Read from File or Write to File stages that use the same input file or a file that has the same traits as the file you are working with now. You can also use the settings file with job executor to specify file settings at runtime.</p> <p>For information about the settings file, see The File Definition Settings File on page 318.</p>

Fields Tab

The Fields tab defines the names, positions, and, for fixed width and line sequential files, lengths of fields in the file. For more information, see these topics:

[Defining Fields In a Delimited Input File](#) on page 312

[Defining Fields In a Line Sequential or Fixed Width File](#) on page 315

Sort Fields Tab

The Sort Fields tab defines fields by which to sort the input records before they are sent into the dataflow. Sorting is optional. For more information, see [Sorting Input Records](#) on page 317.

Runtime Tab

Field Name	Description
File name	Displays the file name selected in the first tab.
Starting record	If you want to skip records at the beginning of the file when reading records into the dataflow, specify the first record you want to read. For example, if you want to skip the first 50 records, in a file, specify 51. The 51st record will be the first record read into the dataflow.
All records	Select this option if you want to read all records starting from the record specified in the Starting record field to the end of the file.
Max records	Select this option if you want to only read in a certain number of records starting from the record specified in the Starting record field. For example, if you want to read the first 100 records, select this option and enter 100.

Defining Fields In a Delimited Input File

The **Fields** tab defines the names, position, and, for some file types, lengths, of the fields in the file. After you define an input file on the **File Properties** tab you can define the fields.

If the input file does not contain a header record, or if you want to manually define the fields, follow these steps on the **Fields** tab:

1. To define the fields already present in the input file, click **Regenerate**. Then, click **Detect Type**. This will automatically set the data type for each field based on the first 50 records in the file.
2. To add additional fields in the output, click **Add**.

3. In the **Name** field, choose the field you want to add or type the name of the field.
4. In the **Type** field, you can leave the data type as `string` if you do not intend to perform any mathematical or date time operations with the data. However, if you intend to perform these kinds of operations, select an appropriate data type. This will convert the string data from the file to a data type that will enable the proper manipulation of the data in the dataflow.

Spectrum Technology Platform supports these data types:

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.

bytearray An array (list) of bytes.

Note: Bytearray is not supported as an input for a REST service.

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 $(2-2^{-52}) \times 2^{1023}$. 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 $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -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).

list Strictly speaking, a list is not a data type. However, when a field contains hierarchical data, it is treated as a "list" field. In Spectrum Technology Platform a list is a collection of data consisting of multiple values. For example, a field Names may contain a list of name values. This may be represented in an XML structure as:

```
<Names>
  <Name>John Smith</Name>
  <Name>Ann Fowler</Name>
</Names>
```

It is important to note that the Spectrum Technology Platform list data type is different from the XML schema list data type in that the XML list data type is a

simple data type consisting of multiple values, whereas the Spectrum Technology Platform list data type is similar to an XML complex data type.

- 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.

5. If you selected a date, time, or numeric data type, you can use the default date and time or number format or you can specify a different format for this specific field. The default format is either the system default format that has been set in the type conversion options in Spectrum Management Console, or it is the dataflow's default format specified in the type conversion options in Spectrum Enterprise Designer. The format that is in effect is displayed. To use the default format, leave **Default** selected. To specify a different format, choose **Custom** and follow these steps:

Note: It is important that you choose a date and time format that accurately reflects the data you are reading from the file. For example, if the file contains date data in the format Month/Day/Year but you choose Day/Month/Year, any date calculations you perform in the dataflow, such as sorting by date, will not reflect the correct date. In addition, records may fail type conversion, in which case the failure behavior specified in the type conversion options in Spectrum Management Console or Spectrum Enterprise Designer will take effect.

- a) In the **Locale** field, select the country whose formatting convention you want to use. Your selection will determine the default values in the **Format** field. For date data, 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 **Format** field, select the format for the data. The format depends on the data type of the field. A list of the most commonly used formats for the selected locale is provided.

An example of the selected format is displayed to the right of the **Format** field.

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 32. To specify your own number format, type the format into the file using the notation described in [Number Patterns](#) on page 34.

6. In the **Position** field, enter the position of this field within the record.

For example, in this input file, AddressLine1 is in position 1, City is in position 2, StateProvince is in position 3, and PostalCode is in position 4.

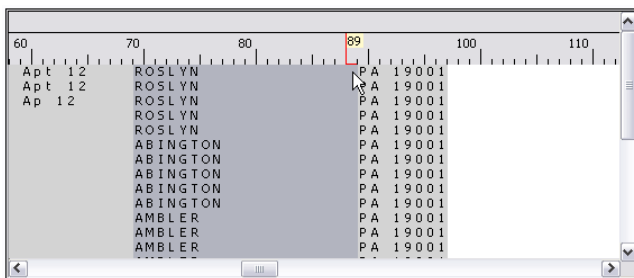
```
"AddressLine1"|"City"|"StateProvince"|"PostalCode"
"7200 13TH ST"|"MIAMI"|"FL"|"33144"
"One Global View"|"Troy"|"NY"|12180
```

7. If you want to have any excess space characters removed from the beginning and end of a field's value string, select the **Trim** check box.

Defining Fields In a Line Sequential or Fixed Width File

In the Read from File stage, the **Fields** tab defines the names, position, and, for some file types, lengths, of the fields in the file. After you define an input file on the **File Properties** tab you can define the fields.

1. On the **Fields** tab, under **Preview**, click at the beginning of a field and drag to the left so that the desired field is highlighted, as shown in the following image:



2. In the **Name** field, enter the field you want to add.
3. In the **Type** field, you can leave the data type as string if you do not intend to perform any mathematical or date and time operations with the data. However, if you intend to perform these kinds of operations, select an appropriate data type. This will convert the string data from the file to a data type that will enable the proper manipulation of the data in the dataflow.

Spectrum Technology Platform supports these data types:

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.

bytearray An array (list) of bytes.

Note: Bytearray is not supported as an input for a REST service.

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 $(2-2^{-52}) \times 2^{1023}$. 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 $(2-2^{-23}) \times 2^{127}$. 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).
list	Strictly speaking, a list is not a data type. However, when a field contains hierarchical data, it is treated as a "list" field. In Spectrum Technology Platform a list is a collection of data consisting of multiple values. For example, a field Names may contain a list of name values. This may be represented in an XML structure as:
	<pre><Names> <Name>John Smith</Name> <Name>Ann Fowler</Name> </Names></pre>
	It is important to note that the Spectrum Technology Platform list data type different from the XML schema list data type in that the XML list data type is a simple data type consisting of multiple values, whereas the Spectrum Technology Platform list data type is similar to an XML complex data type.
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.

- To apply the `packed decimal` storage format to the field, check the **Packed Decimal** checkbox. The `packed decimal` type uses 4 bits as compared to the `integer` type which uses 8 bits.

Note: This storage format is available only on selecting the datatypes `double`, `integer` and `long` while reading line sequential and fixed width files.

- If you selected a date, time, or numeric data type, you can use the default date and time or number format or you can specify a different format for this specific field. The default format is either the system default format that has been set in the type conversion options in Spectrum Management Console, or it is the dataflow's default format specified in the type conversion options in Spectrum Enterprise Designer. The format that is in effect is displayed. To use the

default format, leave **Default** selected. To specify a different format, choose **Custom** and follow these steps:

Note: It is important that you choose a date and time format that accurately reflects the data you are reading from the file. For example, if the file contains date data in the format Month/Day/Year but you choose Day/Month/Year, any date calculations you perform in the dataflow, such as sorting by date, will not reflect the correct date. In addition, records may fail type conversion, in which case the failure behavior specified in the type conversion options in Spectrum Management Console or Spectrum Enterprise Designer will take effect.

- a) In the **Locale** field, select the country whose formatting convention you want to use. Your selection will determine the default values in the **Format** field. For date data, 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 **Format** field, select the format for the data. The format depends on the data type of the field. A list of the most commonly used formats for the selected locale is provided.

An example of the selected format is displayed to the right of the **Format** field.

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 32. To specify your own number format, type the format into the file using the notation described in [Number Patterns](#) on page 34.

6. The **Start Position** and **Length** fields are automatically filled in based on the selection you made in the file preview.
7. If you want to have any excess space characters removed from the beginning and end of a field's character string, select the **Trim Spaces** check box.
8. Click **OK**.

Sorting Input Records

In the Read from File stage, the **Sort Fields** tab defines fields by which to sort the input records before they are sent into the dataflow. Sorting is optional.

1. On the **Sort Fields** tab, click **Add**.
2. Click the drop-down arrow in the **Field Name** column and select the field you want to sort by. The fields available for selection depend on the fields defined in this input file.
3. In the **Order** column, select Ascending or Descending.
4. Repeat until you have added all the input fields you wish to use for sorting. Change the order of the sort by highlighting the row for the field you wish to move and clicking **Up** or **Down**.
5. Default sort performance options for your system are set in Spectrum Management Console. If you want to override your system's default sort performance options, click **Advanced**. The **Advanced Options** dialog box contains these sort performance options:

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:

$$\text{NumberOfRecords} \times 2 \div \text{InMemoryRecordLimit} = \text{NumberOfTempFiles}$$

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: $(\text{InMemoryRecordLimit} \times \text{MaxNumberOfTempFiles} \div 2) \geq \text{TotalNumberOfRecords}$

The File Definition Settings File

A file definition settings file contains the file layout, encoding, and sort options that have been exported from a Read from File or Write to File stage. The file definitions settings file can be imported into Read from File or Write to File to quickly set the stage's options instead of manually specifying the options.

The easiest way to create a file definition settings file is to use specify the file settings using Read from File or Write to File, then click the **Export** button to generate the file definitions settings file.

However, for your information the schema of the file definition settings file is shown below. Each element in the XML file has a type, and if that type is anything other than string or integer, the acceptable values are shown. These values correspond directly to options in the stage's dialog box.

For example, the `FileTypeEnum` element corresponds to the Record Type field on the File Properties tab, and these values appear in the schema: `linesequential`, `fixedwidth`, and `delimited`.

Note: If you enter "custom" for the `LineSeparator`, `FieldSeparator` or `TextQualifier` fields, a corresponding custom element must also be included (for example, "CustomLineSeparator", "CustomFieldSeparator", or "CustomTextQualifier") with a hexadecimal number representing the character, or sequence of characters, to use.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="FileSchema" nillable="true" type="FileSchema"/>
  <xs:complexType name="FileSchema">
    <xs:sequence>
      <xs:element
        minOccurs="0"
        maxOccurs="1"
        default="linesequential"
        name="Type"
        type="FileTypeEnum"/>
      <xs:element
        minOccurs="0"
        maxOccurs="1"
        default="UTF-8" name="Encoding" type="xs:string"/>
      <xs:element
        minOccurs="0"
        maxOccurs="1"
        name="RecordLength"
        type="xs:int"/>
      <xs:element
        minOccurs="0"
        maxOccurs="1"
        default="default"
        name="LineSeparator"
        type="LineSeparatorEnum"/>
      <xs:element
        minOccurs="0"
        maxOccurs="1"
        name="CustomLineSeparator"
        type="xs:string"/>
      <xs:element
        minOccurs="0"
        maxOccurs="1"
        default="comma"
        name="FieldSeparator"
        type="FieldSeparatorEnum"/>
      <xs:element
        minOccurs="0"
        maxOccurs="1"
        name="CustomFieldSeparator"
        type="xs:string"/>
      <xs:element
```

```

        minOccurs="0"
        maxOccurs="1"
        default="none"
        name="TextQualifier"
        type="TextQualifierEnum"/>
<xs:element
    minOccurs="0"
    maxOccurs="1"
    name="CustomTextQualifier"
    type="xs:string"/>
<xs:element
    minOccurs="0"
    maxOccurs="1"
    default="false"
    name="HasHeader"
    type="xs:boolean"/>
<xs:element
    minOccurs="0"
    maxOccurs="1"
    default="true"
    name="EnforceColumnCount"
    type="xs:boolean"/>
<xs:element
    minOccurs="0"
    maxOccurs="1"
    name="Fields"
    type="ArrayOfFieldSchema"/>
</xs:sequence>
</xs:complexType>
<xs:simpleType name="FileTypeEnum">
    <xs:restriction base="xs:string">
        <xs:enumeration value="linesequential"/>
        <xs:enumeration value="fixedwidth"/>
        <xs:enumeration value="delimited"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="LineSeparatorEnum">
    <xs:restriction base="xs:string">
        <xs:enumeration value="default"/>
        <xs:enumeration value="windows"/>
        <xs:enumeration value="linux"/>
        <xs:enumeration value="mac"/>
        <xs:enumeration value="custom"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="FieldSeparatorEnum">
    <xs:restriction base="xs:string">
        <xs:enumeration value="comma"/>
        <xs:enumeration value="tab"/>
        <xs:enumeration value="space"/>
        <xs:enumeration value="semicolon"/>
        <xs:enumeration value="period"/>
        <xs:enumeration value="pipe"/>
    </xs:restriction>
</xs:simpleType>

```



```

    <xs:enumeration value="custom"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="TextQualifierEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="none"/>
    <xs:enumeration value="single"/>
    <xs:enumeration value="double"/>
    <xs:enumeration value="custom"/>
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="ArrayOfFieldSchema">
  <xs:sequence>
    <xs:element
      minOccurs="0"
      maxOccurs="unbounded"
      name="Field"
      nillable="true"
      type="FieldSchema"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="FieldSchema">
  <xs:sequence>
    <xs:element
      minOccurs="0"
      maxOccurs="1"
      name="Name"
      type="xs:string"/>
    <xs:element
      minOccurs="0"
      maxOccurs="1"
      default="string"
      name="Type"
      type="xs:string"/>
    <xs:element
      minOccurs="1"
      maxOccurs="1"
      name="Position"
      type="xs:int"/>
    <xs:element
      minOccurs="0"
      maxOccurs="1"
      name="Length"
      type="xs:int"/>
    <xs:element
      minOccurs="0"
      maxOccurs="1"
      default="false"
      name="Trim"
      type="xs:boolean"/>
    <xs:element
      minOccurs="0"
      maxOccurs="1"

```

```

        name="Locale"
        type="Locale"/>
    <xs:element
        minOccurs="0"
        maxOccurs="1"
        name="Pattern"
        type="xs:string"/>
    <xs:element
        minOccurs="0"
        maxOccurs="1"
        default="none"
        name="Order"
        type="SortOrderEnum"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="Locale">
    <xs:sequence>
        <xs:element
            minOccurs="0"
            maxOccurs="1"
            name="Country"
            type="xs:string"/>
        <xs:element
            minOccurs="0"
            maxOccurs="1"
            name="Language"
            type="xs:string"/>
        <xs:element
            minOccurs="0"
            maxOccurs="1"
            name="Variant"
            type="xs:string"/>
    </xs:sequence>
</xs:complexType>
<xs:simpleType name="SortOrderEnum">
    <xs:restriction base="xs:string">
        <xs:enumeration value="none"/>
        <xs:enumeration value="ascending"/>
        <xs:enumeration value="descending"/>
    </xs:restriction>
</xs:simpleType>
</xs:schema>

```

Configuring Dataflow Options

This procedure describes how to configure a dataflow to support runtime options for Read from File stage.

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. Expand the **Read from File** stage.
The Dataflow options exposed are:
 - a. Character Encoding
 - b. Field Separator
 - c. Text Qualifier
 - d. Record Length
 - e. First Row is Header Record
 - f. Starting Record
 - g. Max Records
6. The selected Read from File option name is displayed in **Option name** and **Option label** fields. This is the option name that will have to be specified at run time in order to set this option.
7. Enter a description of the option in the **Description** field.
8. In the **Target** field, select the option **Selected stage(s)**.
9. 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.
10. 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 845.
11. Click **OK**.
12. Continue adding options as desired.
13. Click **OK** in the **Dataflow Options** dialog box when you are done adding options.
14. Save and expose the dataflow.

Dataflow Options Rules

1. *Character Encoding*: All encoding types that are valid for the underlying JVM are accepted. This option cannot be empty.
2. *Field Separator*: Any single character delimiter is accepted. Currently, HEX values and spaces are not supported.
3. *Text Qualifier*: Any single character qualifier is accepted. HEX values are not supported.
4. *Record Length*: Only integers accepted. Cannot be blank or non numeric.
5. *Starting Record*: Only integers accepted. Cannot be non numeric.
6. *Max records*: Only integers accepted. Cannot be non numeric.
7. *First Row is Header*: Only boolean values of `true` and `false` accepted. Cannot be blank.

Read from Hadoop Sequence File

The **Read from Hadoop Sequence File** stage reads data from a sequence file as input to a dataflow. A sequence file is a flat file consisting of binary key/value pairs. For more information, refer to <https://knpcode.com/hadoop/hadoop-io/how-to-read-and-write-sequencefile-in-hadoop/>.

The stage supports reading data from and writing data to HDFS 3.x. The support includes:

- Connectivity to HDFS from Spectrum on Windows
- Support and connectivity to Hadoop 3.x from Spectrum with high availability
- Kerberos-enabled HDFS connectivity through Windows

Also see [Configuring HDFS Connection for HA Cluster](#) and [Best Practices for connecting to HDFS 3.x and Hive 2.1.1](#).

Note: The **Read from Hadoop Sequence File** stage only supports delimited, uncompressed sequence files located on **Hadoop Distributed File System (HDFS)**.

Related tasks:

Connecting to Hadoop: To be able to read a file located on the Hadoop system or to write a file to it, you need to create a connection to the Hadoop file server. Once you do that, the name by which you save the connection is displayed as the server name.

File Properties Tab

Fields	Description
Server	<p>Indicates the file you select in the File name field is located on the Hadoop system.</p> <p>Note: You need to create a connection to the Hadoop file server before using it here. For details on creating connection, see Connecting to Hadoop.</p> <p>If you select a file on the Hadoop system, the server name will be the name you specify while creating a file server.</p>
File name	Specifies the path to the file. Click the ellipses button (...) to go to the file you want.

Fields	Description
Field separator	<p>Specifies the character used to separate fields in a delimited file. For example, this record uses a pipe () as a field separator:</p> <pre>7200 13TH ST MIAMI FL 33144</pre> <p>These characters available to define as field separators are:</p> <ul style="list-style-type: none"> • Space • Tab • Comma • Period • Semicolon • Pipe <p>If the file uses a different character as a field separator, click the ellipses button to select another character as a delimiter.</p>
Text qualifier	<p>The character used to surround text values in a delimited file. For example, this record uses double quotes (") as a text qualifier.</p> <pre>"7200 13TH ST" "MIAMI" "FL" "33144"</pre> <p>The characters available to define as text qualifiers are:</p> <ul style="list-style-type: none"> • Single quote (') • Double quote (") <p>If the file uses a different text qualifier, click the ellipses button to select another character as a text qualifier.</p>

Fields Tab

The Fields tab defines the names, positions, and types of fields in the file. For more information, see [Defining Fields In an Input Sequence File](#) on page 326.

Sort Fields Tab

The Sort Fields tab defines fields by which to sort the input records before they are sent into the dataflow. Sorting is optional. For more information, see [Sorting Input Records](#) on page 326.

Filter Tab

The Field tab defines fields by which to filter the input records before they are sent into the dataflow. For more information, see [Filtering Input Records](#) on page 327.

Defining Fields In an Input Sequence File

In the Read from Hadoop Sequence File stage, the **Fields** tab defines the names, positions, and types of fields in the file. After you define an input file on the **File Properties** tab you can define the fields.

The **Fields** tab defines the names, position, and, for some file types, lengths, of the fields in the file. After you define an input file on the **File Properties** tab you can define the fields.

1. To add additional fields in the output, click **Add**.
2. In the **Type** field, you can leave the data type as string if you do not intend to perform any mathematical operations with the data. However, if you intend to perform these kinds of operations, select an appropriate data type. This will convert the string data from the file to a data type that will enable the proper manipulation of the data in the dataflow.

The stage supports the following data types:

double	A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. 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 $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -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.

3. In the **Name** field, choose the field you want to add or type the name of the field.

Sorting Input Records

In the Read from Hadoop Sequence File stage, the **Sort Fields** tab defines fields by which to sort the input records before they are sent into the dataflow. Sorting is optional.

1. In Read from Hadoop Sequence File, click the **Sort Fields** tab.
2. On the **Sort Fields** tab, click **Add**.
3. Click the drop-down arrow in the **Field Name** column and select the field you want to sort by. The fields available for selection depend on the fields defined in this input file.
4. In the **Order** column, select Ascending or Descending.
5. Repeat until you have added all the input fields you wish to use for sorting. Change the order of the sort by highlighting the row for the field you wish to move and clicking **Up** or **Down**.

Filtering Input Records

In the Read from Hadoop Sequence File stage, the **Filter** tab defines fields by which to filter the input records before they are sent into the dataflow. Filtering is optional.

1. In Read from Hadoop Sequence File, click the **Filter** tab.
2. In the **Combine expression method** field, choose **All** if you want all the expressions to evaluate to true in order for the record to be routed to this port; select **Any** if you want records to be routed to this port if one or more of the expressions is true.
3. Click **Add**, specify the field to test, the operator, and a value. The operators are listed in the table, below.

Operator	Description
Is Equal	Checks if the value in the field matches the value specified.
Is Not Equal	Checks if the value in the field does not match the value specified.
Is Greater Than	Checks if the field has a numeric value that is greater than the value specified. This operator works on numeric data types as well as string fields that contain numbers.
Is Greater Than Or Equal To	Checks if the field has a numeric value that is greater than or equal to the value specified. This operator works on numeric data types as well as string fields that contain numbers.
Is Less Than	Checks if the field has a numeric value that is less than the value specified. This operator works on numeric data types as well as string fields that contain numbers.
Is Less Than Or Equal To	Checks if the field has a numeric value that is less than or equal to the value specified. This operator works on numeric data types as well as string fields that contain numbers.
Is Null	Checks if the field is a null value.
Is Not Null	Checks if the field is not a null value.

4. Select the **Trim** option as desired. This option, first trims all the white spaces that may be present before and after the value of the field, before filtering the data in the field.
5. Repeat until you have added all the input fields you wish to use for filtering.

Read From Hive File

The **Read from Hive File** stage reads data from the selected file, which can be in any of these formats: ORC, Parquet, and Avro.

The stage supports reading data from and writing data to HDFS 3.x. The support includes:

- Connectivity to HDFS and Hive from Spectrum on Windows
- Support and connectivity to Hadoop 3.x from Spectrum with high availability
- Kerberos-enabled HDFS connectivity through Windows
- Support of Datetime datatype in the Parquet file format

Also see [Configuring HDFS Connection for HA Cluster](#) and [Best Practices for connecting to HDFS 3.x and Hive 2.1.1](#).

Related task:

Connecting to Hadoop: To be able to use **Read from Hive File** stage, you need to create a connection to the Hadoop file server. Once you do that, the name by which you save the connection is displayed as the server name.

File Properties tab

Fields	Description
Server	<p>Indicates the file you select in the File name field is located on the Hadoop system.</p> <p>Note: You need to create a connection to the Hadoop file server before using it here. For details on creating connection, see Connecting to Hadoop.</p> <p>If you select a file on the Hadoop system, the server name will be the name you specify while creating a file server.</p>
File name	<p>Specifies the path to the file. Click the ellipses button (...) to browse to the file you want. You may, however, rename the columns of the schema as required. The first 50 records of the file are fetched in the Preview grid on selecting the file.</p> <p>Note: The schema of an input file is imported as soon as you go to the correct location and select the file. This imported schema cannot be edited.</p>
File type	<p>Select the type of the file being read:</p> <ul style="list-style-type: none"> • ORC • Parquet • Avro

Fields tab

The **Fields** tab defines the names, datatypes, positions of the fields as present in the input file, as well as the user-given names for the fields. For more information, see [Defining Fields for Reading from Hive File](#) on page 329.

Defining Fields for Reading from Hive File

In the **Fields** tab of the **Read from Hive File** stage, the schema names, datatypes, positions, and the given names of the fields in the file are listed.

1. Click **Regenerate**.

For ORC, Avro, and Parquet files, this generates the schema based on the metadata of the existing file.

The grid displays the columns **Name**, **Type**, **Stage Field**, and **Include**.

The **Name** column displays the field name, as derived from the header record of the file.

The **Type** column lists the datatypes of each respective field of the file.

The stage supports these data types:

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. Note: The <code>datetime</code> datatype in Spectrum maps to the <code>timestamp</code> datatype of Hive files.
double	A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2 \cdot 2^{-52}) \times 2^{1023}$. In E notation, the range of values is -1.79769313486232E+308 to 1.79769313486232E+308.
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 <code>bigdecimal</code> data type supports more precise calculations than the <code>double</code> data type. Note: For Avro and Parquet Hive files, fields of the <code>decimal</code> datatype in the input file are converted to <code>bigdecimal</code> datatype.

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).

Note: The `long` datatype in Spectrum maps to the `bigint` datatype of Hive files.

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).

float A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{23})\times 2^{127}$. In E notation, the range of values -3.402823E+38 to 3.402823E+38.

string A sequence of characters.

The **Position** column displays the starting position of the respective field within a record.

- In the **Stage Field** column, edit the existing field name to the desired name for each field. By default, this column displays the field names read from the file.
- In the **Include** column, select the checkboxes against the fields you wish to include in the output of the stage. By default, all the fields are selected in this column.
- Click **OK**.

Read from HL7 File

The **Read from HL7 File** stage reads Health Level Seven (HL7) data from a text file as input to a dataflow. HL7 is a messaging standard used in the healthcare industry to exchange data between systems. For more information about HL7, visit <http://www.hl7.org>.

HL7 Message Format

Data in an HL7 message is organized hierarchically as follows:

```
message
  segment
    field
      component
        subcomponent
```

Each line of an HL7 message is a segment. A *segment* is a logical grouping of fields. The first three characters in a segment identify the segment type. In the above message, there are five segments MSH (message header), PID (patient identification), two NK1 (next of kin) segments, and IN1 (insurance).

Each segment consists of fields. A *field* contains information related to the purpose of the segment, such as the name of the insurance company in the IN1 (insurance) segment. Fields are typically (but not always) delimited by a | character.

Fields can be divided into *components*. Components are typically indicated by a ^ character. In the above example, the PID (patient identification) segment contains a patient name field containing LEVERKUHN^ADRIAN^C which has three parts, last name (LEVERKUHN), first name (ADRIAN), and middle initial (C). Components can be divided into *subcomponents*. Subcomponents are typically indicated by a & character.

This is an example of an HL7 message:

```
MSH|^~\&||.||||199908180016||ADT^A04|ADT.1.1698593|P|2.7
PID|1||000395122||LEVERKUHN^ADRIAN^C||19880517180606|M|||6 66TH AVE
NE^^WEIMAR^DL^98052|| (157) 983-3296|||S||12354768|87654321
NK1|1|TALLIS^THOMAS^C|GRANDFATHER|12914 SPEM
ST^ALIUM^IN^98052| (157) 883-6176
NK1|2|WEBERN^ANTON|SON|12 STRASSE MUSIK^^VIENNA^AUS^11212| (123) 456-7890
IN1|1|PRE2||LIFE PRUDENT BUYER|PO BOX
23523^WELLINGTON^ON^98111|||19601|||||THOMAS^JAMES^M|F|ZKA535529776
```

Note: To create an HL7 file using the given sample text:

1. Copy and paste the sample text into a new document using any text editing software.
2. Make the required content changes.
3. Configure the settings to view EOL (End of Line) on the text. In your text editor, go to **View > Show Symbol > Show End of Line**.
4. Change the EOL (End of Line) Conversion format to CR (Carriage Return). In your text editor, go to **Edit > EOL Conversion > Old Mac Format**.
5. Save the HL7 file after making this format change.

File Properties Tab

Field Name	Description
Server name	Indicates whether the file you select as input is located on the computer running Spectrum Enterprise Designer or on the Spectrum Technology Platform server. If you select a file on the local computer, the server name will be My Computer. If you select a file on the server the server name will be Spectrum Technology Platform.
File name	Specifies the path to the file. Click the ellipses button (...) to access the file you want. Note: If the Spectrum Technology Platform server is running on Linux, remember that file names and paths on these platforms are case sensitive.

Field Name	Description
HL7 Version	The version of the HL7 standard used in the file you specified. For example, "2.7" means that the file uses HL7 version 2.7. The HL7 version is indicated by the 12th field in the <code>MSH</code> segment.
Character encoding	<p>The text file's encoding. Select one of these:</p> <p>CP1252 This encoding is also known as the Windows-1252 or simply Windows character set. It is a super set of ISO-8859-1 and uses the 128-159 code range to display additional characters not included in the ISO-8859-1 character set.</p> <p>UTF-8 Supports all Unicode characters and is backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html.</p> <p>UTF-16 Supports all Unicode characters but is not backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html.</p> <p>US-ASCII A character encoding based on the order of the English alphabet.</p> <p>UTF-16BE UTF-16 encoding with big endian byte serialization (most significant byte first).</p> <p>UTF-16LE UTF-16 encoding with little endian byte serialization (least significant byte first).</p> <p>ISO-8859-1 An ASCII character encoding typically used for Western European languages. Also known as Latin-1.</p> <p>ISO-8859-3 An ASCII character encoding typically used for Southern European languages. Also known as Latin-3.</p> <p>ISO-8859-9 An ASCII character encoding typically used for Turkish language. Also known as Latin-5.</p> <p>CP850 An ASCII code page used to write Western European languages.</p> <p>CP500 An EBCDIC code page used to write Western European languages.</p> <p>Shift_JIS A character encoding for the Japanese language.</p> <p>MS932 A Microsoft's extension of Shift_JIS to include NEC special characters, NEC selection of IBM extensions, and IBM extensions.</p> <p>CP1047 An EBCDIC code page with the full Latin-1 character set.</p>

Field Name	Description
Validate	<p>These options specify whether to check the file to ensure that it conforms to the HL7 2.7 standard. If any message in the file fails validation, it is treated as a malformed record and the malformed record options specified for the job (in Spectrum Enterprise Designer under Edit > Job Options) or for the system (in Spectrum Management Console) will take effect.</p> <p>Required fields Check this box if you want to make sure that each segment, field, component, and subcomponent contains the elements that are required by the HL7 2.7 standard.</p> <p>Length Check this box if you want to make sure that each element meets the minimum and maximum length requirements for the element as defined in the HL7 2.7 standard.</p>

Field Name	Description
------------	-------------

Ignore unexpected	Select these options if you want to allow messages to contain segments, fields, components, and subcomponents that are not in the expected position. The expected positions are defined in the HL7 standard or, in the case of custom message types, in the HL7 Schema Management tool in Spectrum Enterprise Designer.
-------------------	---

For example, consider this custom message schema:

```
MSH
[PID]
{ZSS}
PV1
NK1
{[DG1]}
```

And this data:

```
MSH|^~\&|Pharm|GenHosp|CIS|GenHosp|198807050000||RAS^O17|RAS1234|P|2.7
ZSS|100|abc
PID|1234|||PATID1234^5^M11^ADT1^MR^GOOD HEALTH
HOSPITAL~123456789^^^USSSA^SS|
PV1||O|O/R|||0148^ADDISON, JAMES|0148^ADDISON, JAMES
NK1|Son|John
```

In this case the segment `PID` is unexpected because it is before the `ZSS` segment.

Messages that contain elements in unexpected positions are treated as malformed records and the malformed record options specified for the job (in Spectrum Enterprise Designer under **Edit > Job Options**) or for the system (in Spectrum Management Console) take effect.

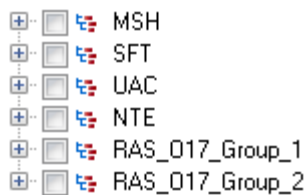
By default all the **Ignore unexpected** options are enabled to allow as many records as possible to be processed successfully.

- Segments** Check this box to allow messages to contain segments that are not defined in the HL7 2.7 standard. Unexpected segments are ignored and other segments in the message are processed.
- Fields** Check this box to allow segments to contain fields that are not defined in the HL7 2.7 standard. Unexpected fields are ignored and other fields in the segment are processed.
- Components** Check this box to allow fields to contain components that are not defined in the HL7 2.7 standard. Unexpected components are ignored and other components in the field are processed.
- Subcomponents** Check this box to allow components to contain subcomponents that are not defined in the HL7 2.7 standard. Unexpected subcomponents in the component are ignored and other subcomponents are processed.

Fields Tab

The **Fields** tab displays the segments, fields, components, and subcomponents. Use the **Fields** tab to select the data you want to read into the dataflow.

Segment groups, which are collections of segments that are used together to contain a category of data, are displayed using a numbering system that shows where in the message schema the group appears. Each segment group is given a label "Group_n" where "n" is a number that corresponds to the group's position in the message schema. To illustrate how the number system works, consider this example:

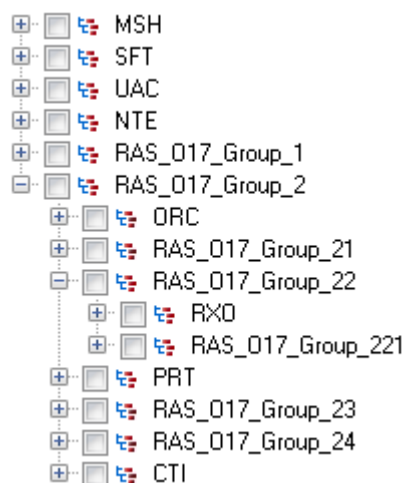


This example shows the field list for the message RAS^017. This message has two segment groups: RAS_017_Group_1 and RAS_017_Group_2. The "Group_1" segment group refers to the first segment group in the RAS^017 schema and the second group, "Group_2", refers to the second group listed in the RAS^017 schema.

To determine which segment group is represented by "Group_1" and "Group_2", find the description of the message RAS^017 in the document *HL7 Version 2.7 Standard*. You can download a copy of this document from <http://www.hl7.org>.

In the description of the message, find the first group, which in the case of RAS^017 is the PATIENT group. The second group in the schema is the ORDER group.

Segment groups that are nested under a segment group have an additional number appended to their group number. For example, Group_21 represents the first group nested under the second group. Further subgroups have additional numbers appended to them, such as Group_221, which for message RAS^017 represents the segment group ORDER_DETAIL_SUPPLEMENT. An example of nested groups is shown here:



The controls on the **Fields** tab are described in this table.

Table 28: Fields Tab

Option Name	Description
Regenerate	<p>Click this button to populate the Fields tab with a list of all segments, fields, components, and subcomponents for the message type contained in the input file. All elements for the message type are displayed based on the HL7 schema regardless of whether the input file contains all the elements. For example, if the file contains an RAS message, the schema for the entire RAS message type will be displayed regardless of whether the input file actually contains data for all the segments, fields, components, and subcomponents.</p> <p>If you have defined any custom elements using the HL7 Schema Management tool in Enterprise Designer, those elements are also listed.</p>
Expand All	Expands all elements in the fields tab so you can view all segments, fields, components, and subcomponents of the message types contained in the file.
Collapse All	Closes all nodes in the view so that only the segments are displayed. Use this to easily see the segments for the message types in the file. You can then expand individual segments in order to view the fields, components, and subcomponents in a segment.
Select all	Check this box to create dataflow fields for all segments, fields, components, and subcomponents for all message types included in the file.

Flattening HL7 Data

HL7 data is organized hierarchically. Since many stages require data to be in a flat format, you may have to flatten the data in order to make the data usable by downstream stages for things like address validation or geocoding.

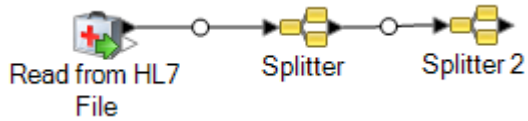
The following procedure describes how to use a Splitter stage to flatten HL7 data.

1. Add a Read from HL7 File stage to your data flow and configure the stage.
2. Add a Splitter stage and connect it to Read from HL7 File.
3. Add additional Splitter stages as needed so that you have one splitter stage for each segment, field, or component that you want to flatten.

Note: You only need to flatten the data that you want to process in a downstream stage. Other data can remain in hierarchical format. For example, if you only want to process address data, you only need to flatten the address data.

4. Connect all the Splitter stages.

You should now have a data flow that looks like this:



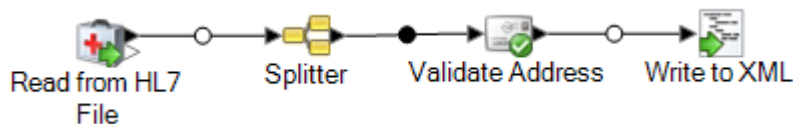
5. Double-click the first Splitter stage to open the stage options.
6. In the **Split at** field, select the segment, field, or component that you want to flatten.
7. Click **OK**.
8. Configure each additional Splitter stage, selecting a different segment, field, or component in each Splitter's **Split at** field.
9. Add additional stages as needed after the last Splitter to complete your dataflow.

Example

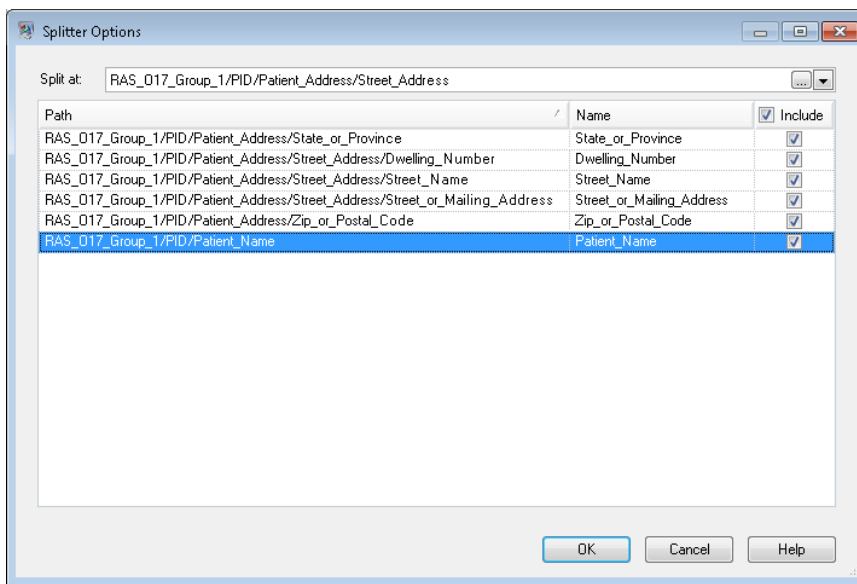
You have the following HL7 data and you want to validate the address in the PID segment.

```
MSH|^~\&||.||||199908180016||RAS^O17|ADT.1.1698594|P|2.7
PID|1||000395122||SMITH^JOHN^D||19880517180606|M|||One Global
View^^Troy^NY^12180||(630)123-4567|||S||12354768|87654321
```

In order to do this you need to convert that address data to flat data so that it can be processed by the Validate Address stage. So, you create a dataflow containing a Splitter followed by a Validate Address stage, like this:



The Splitter stage is configured to split at the PID/Patient_Address/Street_Address component, which converts this data to flat data.



The channel that connects the Splitter stage to the Validate Address stage renames the fields to use the field names required by Validate Address:

Street_or_Mailing_Address is renamed to AddressLine1, State_or_Province is renamed to StateProvince, and Zip_or_Postal_Code is renamed to PostalCode.

In this example, the output is written to an XML file containing this data.

```
<?xml version='1.0' encoding='UTF-8'?>
<XmlRoot xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <PatientInformation>
    <Confidence>95</Confidence>
    <RecordType>Normal</RecordType>
    <CountryLevel>A</CountryLevel>
    <ProcessedBy>USA</ProcessedBy>
    <MatchScore>0</MatchScore>
    <AddressLine1>1 Global Vw</AddressLine1>
    <City>Troy</City>
    <StateProvince>NY</StateProvince>
    <PostalCode>12180-8371</PostalCode>
    <PostalCode.Base>12180</PostalCode.Base>
    <PostalCode.AddOn>8371</PostalCode.AddOn>
    <Country>United States Of America</Country>
    <Patient_Name>
      <Family_Name>
        <Surname>SMITH</Surname>
      </Family_Name>
      <Given_Name>JOHN</Given_Name>
      <Second_and_Further_Given_Names_or_Initials_Thereof>
        D
      </Second_and_Further_Given_Names_or_Initials_Thereof>
    </Patient_Name>
  </PatientInformation>
</XmlRoot>
```

```
</PatientInformation>
</XmlRoot>
```

Adding a Custom HL7 Message

The Read from HL7 File stage validates messages using the HL7 2.7 schema. However, your HL7 data may contain messages that are not part of the HL7 standard. If you want the Read from HL7 File stage to validate your customized HL7 data, you need to create a custom HL7 schema. This topic describes how to create a custom HL7 schema using the HL7 Schema Management tool. For more information about HL7, go to www.hl7.org.

1. In Spectrum Enterprise Designer, go to **Tools > HL7 Schema Management**.

This will open the HL7 Schema Management window which contains a list of supported messages. These messages are predefined by HL7.

2. In the **HL7 Schema Management** window, click **Add**.
3. In the **Message type** field, specify a type for a custom HL7 message.

The message type indicates what health-related information is being provided in the message. For example, an ADT (Admin Discharge Transfer) message type is used to exchange the patient state within a healthcare facility and an ORU (Observation Result) message type is used to transmit observations and results from the LIS (Lab Information System) to the HIS (Hospital Information System).

4. In the **Trigger event** field, specify an event code.

The trigger event is a real-world event that initiates communication and the sending of a message. Both the message type and trigger event are found in the MSH-9 field of the message. For example, MSH-9 field might contain the value ADT^A01. This means that ADT is the HL7 message type, and A01 is the trigger event.

5. In the **Description** field, type a description for a custom HL7 message.

This field helps you to understand more about a message type. For example, if you are adding a XYZ message type, you can provide a description that it is used to exchange the patient state within a healthcare facility.

You will now see a newly created message under the **Definition**. Click on the plus sign to expand the message. You can see the MSH segment is added automatically.

6. To add an existing segment to a message
 - a) Click **Select Segment**.
 - b) Select the segments you want to add to the message and click **OK**.

A schema of a selected segment is displayed in the **Segment Schema** grid and the checked messages are added in the message schema.

7. To add a custom segment to a message

- a) Click **Select Segment**.
- b) Click **Add Segment**.
- c) In the **Name** field, specify a name for the segment and click **OK**.

The newly added segment is displayed at the bottom of the **Segments Supported** list.

- d) Select the added custom segment and then click the **Add field** button.
- e) In the **Name** field, specify a field name of the selected segment.

For example, a PID (Patient information) segment contains field names as Patient ID, Patient Name, Patient Address, County Code, and so on.

- f) In the **Type** field, select an appropriate data type.

HL7 data types define the kind of data that can be included in a field, and are used throughout the HL7 message structure. For example, ST for string, TX for text data and FT for formatted data.

- g) In the **Normative length** field, specify the minimum and maximum length of the field using the following format: m..n. You can also specify a list of possible values for a length of the field using the following format: x,y,z.

For example, length of 1..3 means the length of the item may be either 1, 2 or 3 and the length of 1, 3, 4 means the length of the item may be either 1, 3 or 4 but not 2. A value other than 1, 3, and 4 would be treated as invalid.

- h) In the **Optionality** field, specify whether a field is optional or required.

O

The field is optional.

R

The field is required.

- i) In the **Repetition** field, if you want to allow the field to appear more than once in the segment, check the **Repetition** box and specify the number of times the field can be used.

For example, a value of 3 would mean that the field can have three occurrences. If it is unspecified, there will only be one occurrence, which means this field will not be repeated.

8. Click **OK**.

You can also choose the options **Optional** and **Repeating** from the Segment properties.

9. Choose **Optional** to make the selected segment as optional and choose **Repeating** to allow a repetition of a selected segment in a message.
10. Click **OK**.

The newly added message is displayed at the bottom of the list.

Read from NoSQL DB

The **Read from NoSQL DB** stage reads data from a database table as input to a dataflow. The stage supports MongoDB and Couchbase database (version 5.x and above).

General Tab

Field Name	Description
Connection	<p>Select the required database connection from the dropdown list. The options displayed are based on the connections defined in Spectrum Management Console.</p> <p>To add a new connection, see Connecting to NoSQL.</p> <p>To modify an existing connection, select and open it from the list of connections on the Connections page of Spectrum Management Console, make the required updates, and click the Save button.</p>
Table/View	<p>Specifies the collection or view in the database that you want to query.</p> <p>Note: While the term Table/View is used in the user interface, MongoDB calls it a <i>collection</i>, and Couchbase calls it a <i>view</i>.</p>
Schema File	<p>Click the Browse button (...) to select a JSON Schema file. This file is optional. Fields in the fields tab can be regenerated either using the schema file or database table/view.</p> <p>To clear the selected file path, click Clear.</p> <p>Note: Fields will always be generated using the schema file if one is selected.</p>

Field Name	Description
Where	<p>Enter the required filter criteria, if any, using MongoDB syntax, to fetch specific records. Leave the field blank if no filter criteria is required.</p> <p>This syntax is for a clause with an <i>equal to</i> operator:</p> <pre>{ "<column name>" : "<filter value>" }</pre> <p>You can join multiple clauses using the required operators. For a list of the supported operators in the <code>where</code> clause, see http://docs.mongodb.org/manual/reference/operator/query/.</p> <p>For example, to fetch records where the value of the column <code>customer_name</code> matches the value <code>John</code>, and the value of the column <code>customer_age</code> is greater than or equal to 45, enter the below:</p> <pre>{ \$and: [{ "customer_name": "John" }, { \$gte: ["customer_age", "45"] }] }</pre> <p>Attention: Ensure you do not include the keyword <code>where</code> in this field.</p> <p>Note: Currently, this field is visible only on selecting a MongoDB connection.</p>
Ignore Absent Fields	<p>Fields defined in the schema, if not present in the actual record will not flow to the next stage if this option is selected.</p> <p>Note: If you do not enable this option, fields that are not present in the database table or view, are added and processed with the value as NULL.</p>
Preview	<p>Displays the records from the selected table.</p> <p>Note: For MongoDB data sources, clicking Preview shows the filtered records, if one or more <code>where</code> clauses have been entered in the Where field. If no <code>where</code> clause has been entered, the preview displays all the records.</p> <p>Note: For Couchbase data sources, clicking Preview also displays the added <code>_id</code> field containing the key. If the record already has an <code>_id</code> field, then the added <code>_id</code> field overwrites the pre-existing one at the time of previewing the fields.</p>
Expand All	Expands the items in the preview tree.
Collapse All	Collapses the items in the preview tree.

Fields Tab

The Fields tab allows you to select the data that you want to pass to the next stage. For more information, see [Defining Fields in a NoSQL Database](#) on page 343

Defining Fields in a NoSQL Database

The **Fields** tab displays the field and its type as defined in the NoSQL DB/Schema file.

1. On the **Fields Tab**, click **Regenerate**.

This generates the aggregated data based on the first 50 records. The data is displayed in the following format: `Fieldname (datatype)`.

Note: If the schema file is browsed, the fields are generated using the schema file and the table or view is bypassed. To reset the schema file, click **Clear**.

Note: While reading data from a Couchbase DB, the key of each record is also read. This key is stored as a part of the record using an added `_id` field at the time of regenerating the fields, and is also included in the data sent to the next stage. If the record already has an `_id` field, then this would be overwritten with the added `_id` field at the time of regenerating the fields.

2. To modify the name and type of a field, highlight the field, and click **Modify**.
3. In the **Name** field, choose the field you want to add or type the name of the field.
4. In the **Type** field, you can leave the data type as string if you do not intend to perform any mathematical operations with the data. However, if you intend to perform these kinds of operations, select an appropriate data type. This will convert the string data from the file to a data type that will enable the proper manipulation of the data in the dataflow.

The stage supports the following data types:

double	A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. 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 $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -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.

5. You can also add fields that are not present in the table or schema file. Click **Add** to add a new field. To remove a field, click **Remove**.

Note: You can only add a new field under the List type.

6. Click **OK**.

NoSQL DB Dataflow Options

This procedure describes how to configure a dataflow to support runtime options for NoSQL DB.

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. Expand the NoSQLDB stage.
6. The Dataflow options are exposed as described in the following table:

Database	Read	Write
Mongo DB	Connection	Connection
	Table	Table
Couchbase DB	Connection	Connection
	View	
	Design Document Name	

The selected NoSQL DB option name is displayed in **Option name** and **Option label** fields. This is the option name that will have to be specified at run time in order to set this option.

7. Enter a description of the option in the **Description** field.
8. In the **Target** field, select the option **Selected stage(s)**.
9. 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.
10. 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 845.

11. Click **OK**.
12. Continue adding options as desired.
13. Click **OK** in the **Dataflow Options** dialog box when you are done adding options.
14. Save and expose the dataflow.

Read from SAP

The **Read from SAP** stage reads data from a SAP database as input to a dataflow. It can read data from a single table or multiple tables. When reading data from multiple tables, the stage performs a join operation to determine which records to read into the dataflow.

Table 29: Minimum SAP application requirements

Component	Release	SP Level	Support Package
SAP_BASIS	740	0005	SAPKB74005
SAP_ABA	740	0005	SAPKA74005

Note: The minimum supported version of **SAP Business Suite 7** application is *ECC 6.0 EHP 7* and *CRM 7.0 EHP 3* with its components, release, and support pack levels mentioned in this table.

Connecting to SAP

In order to read data from SAP into a dataflow using Read from SAP, you need to create a connection between Spectrum Technology Platform and your SAP system.

Note: For details on installing the supporting databases and dataflows on the Spectrum Technology Platform server, and configuring the SAP system to communicate with Spectrum Technology Platform see the section *Installing Support Files for Read from SAP* in the *Installation Guide*.

1. Open the SAP connection manager. You can do this in **Spectrum Enterprise Designer** under **Tools > SAP Connection Management** or in the **Read from SAP** stage by clicking the **Manage** button next to the **Connection** field.
2. Click **Add**.
3. In the **Connection name** field, give this connection a name.
4. Complete the other fields with the information about the SAP server you want to connect to. See your SAP Basis administrator for the necessary information.

Important: The user ID and password must be for a SAP account with administrator privileges.

5. Click **Test** to verify the connection.
6. Click **OK**.

You have now created a connection that can be used by the Read from SAP stage to read data from SAP into a dataflow.

Reading Data from a Single SAP Table

The Read from SAP stage can be configured to read data from a single table in the SAP database or multiple tables. This procedure describes how to configure Read from SAP to read data from a single table.

1. In Spectrum Enterprise Designer, drag Read from SAP onto the canvas.
2. Double-click the Read from SAP stage on the canvas.
3. In the **Connection** field, select the SAP server that contains the data you want to read into the dataflow. If there is no connection defined for the SAP server you need to create the connection by clicking **Manage**.
4. In the **Source type** field, choose **Single**.
5. Click **Select**.
6. Select the table you want to read into the dataflow then click **OK**.

Note: Only the first 200 tables are listed. Use the search feature to find tables not listed in the first 200. The search field only searches the values in the **Name** and **Label** columns.

7. To view the field names that will be used in the dataflow, check the box **Display technical name**.

Fields in SAP have a user-friendly name used for display purposes and a unique name that may be less readable. For example, a field may have a user-friendly name of "Distribution Channel" and a technical name of "DIS_CHANNEL". In order to ensure that the field name is valid in the dataflow, the technical name is used as the field name.

8. Check the box in the **Include** column for each field you want to read into the dataflow.
9. Click **OK**.
10. If you want to read only certain records, you can specify filter conditions on the **Filter** tab. In order for a record to be read into the dataflow it must meet all the conditions you define.
11. You can improve performance by specifying an appropriate fetch size on the **Runtime** tab.

Select this option to specify the number of records to read from the database table at a time. For example, if the **Fetch size** value is 100 and total number of records to be read is 1000, then it would take 10 trips to the database to read all the records.

Setting an optimum fetch size can improve performance significantly.

Note: You can calculate an optimum fetch size for your environment by testing the execution times between a Read from DB stage and a Write to Null stage. For more information, see [Determining an Optimum Fetch Size](#).

The default fetch size for Read from SAP is 10,000.

12. Click **OK**.

The Read from SAP stage is now configured to read data from a single table in the SAP database into the dataflow.

Reading Data from Multiple SAP Tables

The Read from SAP stage can be configured to read data from a single table in the SAP database or multiple tables. This procedure describes how to configure Read from SAP to read data from a multiple tables. To read data from multiple tables, you define a JOIN statement to combine data into a single stream.

1. In Spectrum Enterprise Designer, drag Read from SAP onto the canvas.
2. Double-click the Read from SAP stage on the canvas.
3. In the **Connection** field, select the SAP server that contains the data you want to read into the dataflow. If there is no connection defined for the SAP server you need to create the connection by clicking **Manage**.
4. In the **Source type** field, choose **Multiple**.
5. Click **Add**.
6. Select the tables you want to read into the dataflow then click **OK**.

Note: Only the first 200 tables are listed. Use the search feature to find tables not listed in the first 200. The search field only searches the values in the **Name** and **Label** columns.

7. Select the first table in the list and click **Create Relationship**. This is the source table.
8. In the **Source key** field, select the column from the source table whose value will be used match records to records from the other table.
9. In the **Join type** field, select one of the following:

INNER JOIN	Returns only those records that have a match between the source and target tables.
LEFT JOIN	Returns all records from the source table even if there are no matches between the source and target tables. This option returns all records from the source table plus any records that match in the target table.
10. In the **Table** field, select the target table.
11. In the **Table key** field, select the column in the target table containing the data you want to compare to the data from the **Source key** field to determine if the record meets the join condition.
12. Click **OK**.
13. Click **Select Schema**.
14. Choose the fields that you want to read into the dataflow. To view the field names that will be used in the dataflow, check the **Display technical name** box.

Fields in SAP have a user-friendly name used for display purposes and a unique name that may be less readable. For example, a field may have a user-friendly name of "Distribution Channel" and a technical name of "DIS_CHANNEL". In order to ensure that the field name is valid in the dataflow, the technical name is used as the field name.

15. Click **OK**.

16. If you want to read only certain records, you can specify filter conditions on the **Filter** tab. In order for a record to be read into the dataflow it must meet all the conditions you define.
17. You can improve performance by specifying an appropriate fetch size on the **Runtime** tab.

Select this option to specify the number of records to read from the database table at a time. For example, if the **Fetch size** value is 100 and total number of records to be read is 1000, then it would take 10 trips to the database to read all the records.

Setting an optimum fetch size can improve performance significantly.

Note: You can calculate an optimum fetch size for your environment by testing the execution times between a Read from DB stage and a Write to Null stage. For more information, see [Determining an Optimum Fetch Size](#).

The default fetch size for Read from SAP is 10,000.

The Read from SAP stage is now configured to read data from a multiple tables in the SAP database into the dataflow.

Filtering Records in Read from SAP

The filter settings in Read from SAP allow you to read a subset of records from an SAP table rather than all records in the table. To filter records, you specify the values that a record must contain in order for it to be read into the dataflow. If you do not specify any filter conditions, all records in the table are read into the dataflow. Using filter conditions is optional.

Note: If the Read from SAP stage is configured to read data from multiple SAP tables, the filter is applied after the JOIN operation is performed.

1. In the Read from SAP stage, click the **Filter** tab.
2. Click **Add**.
3. In the **Table name** field, select the table that contains the records you want to filter.
4. In the **Filter by** field, select the field that contains the data you want to use as the basis for filtering.
5. Choose one of the following operators:

Note: The operators available to you vary depending on the data type of the field on which you are filtering.

Operator	Description
Contains	Checks if the string contains the value specified.
Equals	Checks if the value in the field matches the value specified.

Operator	Description
Not Equals	Checks if the value in the field does not match the value specified.
Greater Than	Checks if the field has a numeric value that is greater than the value specified. This operator works on numeric data types as well as string fields that contain numbers.
Greater Than Or Equals	Checks if the field has a numeric value that is greater than or equal to the value specified. This operator works on numeric data types as well as string fields that contain numbers.
Less Than	Checks if the field has a numeric value that is less than the value specified. This operator works on numeric data types as well as string fields that contain numbers.
Less Than Or Equals	Checks if the field has a numeric value that is less than or equal to the value specified. This operator works on numeric data types as well as string fields that contain numbers.
Is Null	Checks if the field is a null value.
Is Not Null	Checks if the field is not a null value.
Starts With	Checks if the field starts with the value specified.
Ends With	Checks if the field ends with the value specified.

6. Enter the value to which you want to compare the selected field's value.
7. Click **OK**.
8. Add additional filter conditions if needed.

Note: If you specify multiple filter conditions, all the filter conditions must be true in order for the record to be read into the dataflow. If any one of the conditions is not true, the record is not read into the dataflow.

Read from Spreadsheet

Read from Spreadsheet reads data from an Excel spreadsheet as input to a dataflow in these supported formats: *.xls and *.xlsx.

File Properties Tab

The **File Properties** tab contains options for specifying the spreadsheet and data to read into the dataflow.

Field Name	Description
Server name	Indicates whether the file you select as input is located on the computer running Spectrum Enterprise Designer or on the Spectrum Technology Platform server. If you select a file on the local computer, the server name will be My Computer. If you select a file on the server the server name will be Spectrum Technology Platform.
File name	Specifies the path to the file. Click the ellipses button (...) to locate the file you want. Note: If the Spectrum Technology Platform server is running on Linux, remember that file names and paths on these platforms are case sensitive.
Data selection	Specifies how you want to select data from the spreadsheet to read into the dataflow: <ul style="list-style-type: none"> Sheet Data Select this option to read in all the data from a sheet in the spreadsheet. Range Data Select this option to read in a subset of data from a sheet by specifying a range of cells to read. Named Range Select this option to read in a subset of data from a sheet by specifying a named range from the spreadsheet.
Sheet selection	If you choose Sheet Data or Range Data in the Data selection field, use this option to choose the sheet from which you want to read data into the dataflow.
Range	If you choose Range Data in the Data selection field, use this option to specify the cell that starts the range and the cell that ends the range.
Named range	If you choose Named Range in the Data selection field, use this option to specify the name of the range you want to read into the dataflow. Ranges are defined in the spreadsheet. If no ranges are listed it means that no ranges are defined in the spreadsheet.

Field Name	Description
Header row	<p>Check this box to specify a row that contains column headings. Column headings become the dataflow field names although you can change field names on the Fields tab. If you do not check this box, the dataflow fields are given generic default names such as Column1 and Column2.</p> <p>The header row you specify is relative to the data selection. For example, if you choose Range Data in the Data selection field and the range begins on the fifth row, and you specify 1 as the header row, then the fifth row in the spreadsheet will be used as the header because the fifth row of the spreadsheet is the first row of the range.</p>
Data offset from header	<p>If you specify a header row, this field specifies the first row that contains data, relative to the header. For example, if you specify 1, the first row below the header will be the first row of data to be read into the dataflow. If you specify 2, the second row below the header will be the first row read into the dataflow.</p>
First data row	<p>If you do not specify a header row, this field specifies which row within the data selection contains the first row of data to read into the dataflow. The row you specify is relative to the data selection. For example, if you choose Range Data in the Data selection field and the range begins on the fifth row, and you specify 1 as the first data row, then the first row of data to be read into the dataflow will be the fifth row.</p>
Ignore empty rows	<p>Select this option if you want empty rows to be excluded from the dataflow. If you do not select this option, empty rows in the spreadsheet will result in empty records in the dataflow.</p> <p>Note: This option does not affect the data shown in the preview. Empty rows are always shown in the preview even if this option is selected.</p>

Fields Tab

The **Fields** tab contains options for mapping the data from the spreadsheet to fields in the dataflow.

Option	Description
Regenerate	<p>Click this button to populate the fields tab with the fields in the input file defined on the File Properties tab.</p>

Option	Description
Detect Type	Click this button to automatically determine the data type for all the fields. You can manually change a field's data type by selecting the field and clicking Modify .
Modify	Select a field then click this button to modify the field name or data type.

Read from Variable Format File

Read from Variable Format File reads data from a file containing records of varying layout. Each record is read in as a list field. You can specify the tag that indicates the parent record type, and all other record types will become list fields under the parent.

Variable format files have these characteristics:

- Records in the file may have different fields, and different numbers of fields.
- Each record must contain a tag (usually a number) identifying the type of record.
- Hierarchical relationships are supported.

Example of a Variable Format File

This example shows a variable format file containing information about checking account activity for two customers, Joe Smith and Anne Johnson. In this example, the file is a delimited file that uses a comma as the field delimiter.

```
001 Joe,Smith,M,100 Main St,555-234-1290
100 CHK12904567,12/2/2007,6/1/2012,CHK
200 1000567,1/5/2012,Fashion Shoes,323.12
001 Anne,Johnson,F,1202 Lake St,555-222-4932
100 CHK238193875,1/21/2001,4/12/2012,CHK
200 1000232,3/5/2012,Blue Goose Grocery,132.11
200 1000232,3/8/2012,Trailway Bikes,540.00
```

The first field in each record contains the tag which identifies the type of record and therefore the record's format:

- 001: Customer record
- 100: Account record
- 200: Account transaction record

For delimited files it is common for the tag value (001, 100, 200) to be in a fixed number of bytes at the start of the record as shown in the above example.

Each record has its own format:

- 001: FirstName,LastName,Gender,Address,PhoneNumber
- 100: AccountID,DateOpened,ExpirationDate,TypeOfAccount
- 200: TransactionID,DateOfTransaction,Vendor,Amount

Record format 100 (account record) is a child of the previous 001 record, and record format 200 (account transaction record) is a child of the previous record 100 (account record). So in the example file, Joe Smith's account CHK12904567 had a transaction on 1/5/2012 in the amount of 323.12 at Fashion Shoes. Likewise, Anne Johnson's account CHK238193875 had two transactions, one on 3/5/2012 at Blue Goose Grocery and one on 3/8/2012 at Trailway Bikes.

File Properties Tab

Option Name	Description
Server name	Indicates whether the file you select as input is located on the computer running Spectrum Enterprise Designer or on the Spectrum Technology Platform server. If you select a file on the local computer, the server name will be My Computer. If you select a file on the server the server name will be Spectrum Technology Platform.
File name	<p>Specifies the path to the file. Click the ellipses button (...) to go to the file you want.</p> <p>You can read multiple files by using a wild card character to read data from multiple files in the directory. The wild card characters * and ? are supported. For example, you could specify *.csv to read in all files with a .csv extension located in the directory. In order to successfully read multiple files, each file must have the same layout (the same fields in the same positions). Any record that does not match the layout specified on the Fields tab will be treated as a malformed record.</p> <p>While reading a file from an HDFS file server, the compression formats supported are:</p> <ol style="list-style-type: none"> 1. GZIP (.gz) 2. BZIP2 (.bz2) <p>Note: The extension of the file indicates the compression format to be used to decompress the file.</p> <p>Attention: If the Spectrum Technology Platform server is running on Linux, remember that file names and paths on these platforms are case sensitive.</p>

Option Name	Description
Record type	<p>The format of the records in the file. Select one of:</p> <p>Line Sequential A text file in which records are separated by an end-of-line (EOL) character such as a carriage return or line feed (CR or LF) and each field has a fixed starting and ending character position.</p> <p>Fixed Width A text file in which each record is a specific number of characters in length and each field has a fixed starting and ending character position.</p> <p>Delimited A text file in which records are separated by an end-of-line (EOL) character such as a carriage return or line feed (CR or LF) and each field is separated by a designated character such as a comma.</p>

Option Name	Description
Character encoding	<p>The text file's encoding. Select one of these:</p> <p>CP1252 This encoding is also known as the Windows-1252 or simply Windows character set. It is a super set of ISO-8859-1 and uses the 128-159 code range to display additional characters not included in the ISO-8859-1 character set.</p> <p>UTF-8 Supports all Unicode characters and is backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html.</p> <p>UTF-16 Supports all Unicode characters but is not backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html.</p> <p>US-ASCII A character encoding based on the order of the English alphabet.</p> <p>UTF-16BE UTF-16 encoding with big endian byte serialization (most significant byte first).</p> <p>UTF-16LE UTF-16 encoding with little endian byte serialization (least significant byte first).</p> <p>ISO-8859-1 An ASCII character encoding typically used for Western European languages. Also known as Latin-1.</p> <p>ISO-8859-3 An ASCII character encoding typically used for Southern European languages. Also known as Latin-3.</p> <p>ISO-8859-9 An ASCII character encoding typically used for Turkish language. Also known as Latin-5.</p> <p>CP850 An ASCII code page used to write Western European languages.</p> <p>CP500 An EBCDIC code page used to write Western European languages.</p> <p>Shift_JIS A character encoding for the Japanese language.</p> <p>MS932 A Microsoft's extension of Shift_JIS to include NEC special characters, NEC selection of IBM extensions, and IBM extensions.</p> <p>CP1047 An EBCDIC code page with the full Latin-1 character set.</p>
Record length	For fixed width files, specifies the exact number of characters in each record.

Option Name	Description
Field separator	<p>Specifies the character used to separate fields in a delimited file. For example, this record uses a pipe () as a field separator:</p> <pre data-bbox="565 415 1422 468">7200 13TH ST MIAMI FL 33144</pre> <p>These characters available to define as field separators are:</p> <ul data-bbox="548 535 683 730" style="list-style-type: none"> • Space • Tab • Comma • Period • Semicolon • Pipe <p>If the file uses a different character as a field separator, click the ellipses button to select another character as a delimiter.</p>
Tag separator	<p>Specifies the character placed after the tag field to demarcate the identifying field for each record in a delimited file. A tag separator must be a single character.</p> <p>By default, these characters are available to be selected as tag separators:</p> <ul data-bbox="548 1003 683 1199" style="list-style-type: none"> • Space • Tab • Comma • Period • Semicolon • Pipe <p>If the file uses a different character as a tag separator, click the ellipses button to add and select a custom tag separator.</p> <p>Note: By default, the Record separator character is the same as the selected Field separator character. To enable this field and select a different character, uncheck the Same as Field separator checkbox.</p>
Same as Field separator	<p>Indicates if the tag separator is the same as the field separator. Uncheck this to select a different character as the tag separator.</p> <p>Note: By default, this checkbox is checked and the Tag separator field is disabled.</p>

Option Name	Description
Text qualifier	<p>The character used to surround text values in a delimited file.</p> <p>For example, this record uses double quotes (") as a text qualifier.</p> <pre>"7200 13TH ST" "MIAMI" "FL" "33144"</pre> <p>The characters available to define as text qualifiers are:</p> <ul style="list-style-type: none"> • Single quote (') • Double quote (") <p>If the file uses a different text qualifier, click the ellipses button to select another character as a text qualifier.</p>
Record separator	<p>Specifies the character used to separate records in line a sequential or delimited file. This field is not available if you check the Use default EOL check box.</p> <p>The record separator settings available are:</p> <p>Linux (U+000A) A line feed character separates the records. This is the standard record separator for Linux systems.</p> <p>Macintosh (U+000D) A carriage return character separates the records. This is the standard record separator for Macintosh systems.</p> <p>Windows (U+000D U+000A) A carriage return followed by a line feed separates the records. This is the standard record separator for Windows systems.</p> <p>If your file uses a different record separator, click the ellipses button to select another character as a record separator.</p>
Root tag name	<p>The tag to use for records that are a parent of other record types. For example if you have three record types 001, 100, and 200, and record types 100 and 200 are children of record type 001, then 001 is the root tag.</p>
Use fixed-width tags	<p>Specifies whether to allocate a fixed amount of space at the beginning of each record in which to place the record tag. This example shows a file with the tags 001, 100, and 200 in a fixed-width field:</p> <pre>001 Joe, Smith, M, 100 Main St, 555-234-1290 100 CHK12904567, 12/2/2007, 6/1/2012, CHK 200 1000567, 1/5/2012, Mike 's Shoes, 323.12</pre>
Tag start position	<p>If you check the Use fixed-width tags box, this option specifies the position in each record where the tag begins. For example, if the tag begins in the fourth character in the record, you would specify 4.</p>

Option Name	Description
Tag width	<p>If you check the Use fixed-width tags box, this option specifies the number of spaces to allocate for tags starting from the position specified in the Tag start position field. For example, if you specify 3 in the Tag start position field and you specify 7 in the Tag width field, then positions 4 through 10 would be considered the record tag. The value you specify must be large enough to include all the characters of the longest tag name.</p> <p>The value in the Tag width field is automatically increased if you lengthen the tag name in the Root tag name field.</p> <p>The maximum tag width is 1024.</p>
Use default EOL	<p>Specifies that the file's record separator is the default end of line (EOL) character used on the operating system on which the Spectrum Technology Platform server is running.</p> <p>Do not select this option if the file uses an EOL character that is different from the default EOL character used on the server's operating system. For example, if the file uses a Windows EOL but the server is running on Linux, do not check this option. Instead, select the Windows option in the Record separator field.</p>
Treat records with fewer fields than defined as malformed	<p>If you enable this option, child records that contain fewer fields than a complete record are considered malformed. When a malformed record is encountered, processing advances to the next root tag, ignoring all child tags in between. An exception is written to the log containing information about the malformed child records along with a line number.</p> <p>Records are always considered malformed in these situations, regardless of whether you enable this option.</p> <ul style="list-style-type: none"> • The tag is unknown • The line is empty • There is a tag with no data • A record with a tag that is a child of another tag appears immediately after a record with a root tag

Fields Tab

The **Fields** tab specifies the characteristics of each field read in from the file.

Runtime Tab

Field Name	Description
File name	Displays the file name selected in the first tab.
Starting record	If you want to skip records at the beginning of the file when reading records into the dataflow, specify the first record you want to read. For example, if you want to skip the first 50 records, in a file, specify 51. The 51st record will be the first record read into the dataflow.
All records	Select this option if you want to read all records starting from the record specified in the Starting record field to the end of the file.
Max records	Select this option if you want to only read in a certain number of records starting from the record specified in the Starting record field. For example, if you want to read the first 100 records, select this option and enter 100.

Defining Fields in Delimited Variable Format Files

This procedure describes how to define fields in the Read from Variable Format File stage for delimited files.

1. In the Read from Variable Format File stage, click the **Fields** tab.
2. Click **Regenerate**.

A list of all the fields for each record type is displayed. For each field the following information is displayed:

Parent	The tag from the input file indicating the record type in which the field appears. If the tag begins with a number, the tag is prefixed with "NumericTag_". For example, a tag named 100 would become NumericTag_100. The prefix is necessary because dataflow field names cannot begin with a number.
Field	The name that will be used in the dataflow for the field. By default, fields are given names in the format <Tag Name>_<Column n>. For example, the first field of record type Owner would be Owner_Column1, the second would be Owner_Column2, and so on.
Type	The field's data type.

Note: The first 50 records are used to generate the fields list. The input file must contain at least two root tags in order to generate a fields list.

3. If you want to modify the parent/child relationships between the tags:
 - a) Click **Modify Tag Hierarchy**.
 - b) Click and drag the tags to define the tag hierarchy you want.
 - c) Click **OK**.
4. If you want to modify the a field's name or data type, select the field and click **Modify**.
5. In the **Name** field, choose the field you want to add or type the name of the field.

Typically you will want to replace the default names with meaningful names to represent the data in the field. For example, consider this input data:

```
001    Joe,Smith,M,100 Main St,555-234-1290
```

This record has a parent tag of 001 and would have these fields created by default:

```
NumericTag_001_Column1: Joe
NumericTag_001_Column2: Smith
NumericTag_001_Column3: M
NumericTag_001_Column4: 100 Main St
NumericTag_001_Column5: 555-234-1290
```

You would probably want to rename the fields so that the names describe the data. For example:

```
FirstName: Joe
LastName: Smith
Gender: M
AddressLine1: 100 Main St
PhoneNumber: 555-234-1290
```

Note: You cannot rename list fields. List fields, which contain all the fields for a given record type, always use the tag name from the input file as the field name.

6. To change a field's data type, select the data type you want in the **Type** field.

The following data types are available:

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.

bytearray An array (list) of bytes.

Note: Bytearray is not supported as an input for a REST service.

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 $(2-2^{-52}) \times 2^{1023}$. 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 $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -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).
list	Strictly speaking, a list is not a data type. However, when a field contains hierarchical data, it is treated as a "list" field. In Spectrum Technology Platform a list is a collection of data consisting of multiple values. For example, a field Names may contain a list of name values. This may be represented in an XML structure as: <pre> <Names> <Name>John Smith</Name> <Name>Ann Fowler</Name> </Names> </pre> <p>It is important to note that the Spectrum Technology Platform list data type is different from the XML schema list data type in that the XML list data type is a simple data type consisting of multiple values, whereas the Spectrum Technology Platform list data type is similar to an XML complex data type.</p>
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.

- If you selected a date, time, or numeric data type, you can use the default date and time or number format or you can specify a different format for this specific field. The default format is either the system default format that has been set in the type conversion options in Spectrum Management Console, or it is the dataflow's default format specified in the type conversion options in Spectrum Enterprise Designer. The format that is in effect is displayed. To use the default format, leave **Default** selected. To specify a different format, choose **Custom** and follow these steps:

Note: It is important that you choose a date and time format that accurately reflects the data you are reading from the file. For example, if the file contains date data in the format Month/Day/Year but you choose Day/Month/Year, any date calculations you perform in the dataflow, such as sorting by date, will not reflect the correct date. In addition, records may fail type conversion, in which case the failure behavior specified in the type conversion options in Spectrum Management Console or Spectrum Enterprise Designer will take effect.

- a) In the **Locale** field, select the country whose formatting convention you want to use. Your selection will determine the default values in the **Format** field. For date data, 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 **Format** field, select the format for the data. The format depends on the data type of the field. A list of the most commonly used formats for the selected locale is provided.

An example of the selected format is displayed to the right of the **Format** field.

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 32. To specify your own number format, type the format into the file using the notation described in [Number Patterns](#) on page 34.

8. Click **OK**.

Defining Fields in a Line Sequential or Fixed Width Variable Format File

This procedure describes how to define fields in the Read from Variable Format File stage for line sequential or fixed width files.

1. In the Read from Variable Format File stage, click the **Fields** tab.
2. Click **Get Tags**.

A list of all the fields for each record type is displayed. For each field the following information is displayed:

Parent	The tag from the input file indicating the record type in which the field appears. If the tag begins with a number, the tag is prefixed with "NumericTag_". For example, a tag named 100 would become NumericTag_100. The prefix is necessary because dataflow field names cannot begin with a number.
Field	The name that will be used in the dataflow for the field. By default, fields are given names in the format <Tag Name>_<Column n>. For example, the first field of record type Owner would be Owner_Column1, the second would be Owner_Column2, and so on.
Type	The field's data type.

Note: The first 50 records are used to generate the fields list. The input file must contain at least two root tags in order to generate a fields list.

3. In the **Filter** field, select the tag for the record type whose fields you want to define then click **Add**.

Note: The filter does not have any impact on which fields are read into the dataflow. It only filters the list of fields to make it easier to browse.

4. In the **Name** field, choose the field you want to add or type the name of the field.
5. In the **Type** field, you can leave the data type as `string` if you do not intend to perform any mathematical or date time operations with the data. However, if you intend to perform these kinds of operations, select an appropriate data type. This will convert the string data from the file to a data type that will enable the proper manipulation of the data in the dataflow.

Spectrum Technology Platform supports these data types:

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.

bytearray An array (list) of bytes.

Note: Bytearray is not supported as an input for a REST service.

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 $(2-2^{-52}) \times 2^{1023}$. 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 $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -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).

list Strictly speaking, a list is not a data type. However, when a field contains hierarchical data, it is treated as a "list" field. In Spectrum Technology Platform a list is a collection of data consisting of multiple values. For example, a field

Names may contain a list of name values. This may be represented in an XML structure as:

```
<Names>
  <Name>John Smith</Name>
  <Name>Ann Fowler</Name>
</Names>
```

It is important to note that the Spectrum Technology Platform list data type different from the XML schema list data type in that the XML list data type is a simple data type consisting of multiple values, whereas the Spectrum Technology Platform list data type is similar to an XML complex data type.

- 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.

6. If you selected a date, time, or numeric data type, you can use the default date and time or number format or you can specify a different format for this specific field. The default format is either the system default format that has been set in the type conversion options in Spectrum Management Console, or it is the dataflow's default format specified in the type conversion options in Spectrum Enterprise Designer. The format that is in effect is displayed. To use the default format, leave **Default** selected. To specify a different format, choose **Custom** and follow these steps:

Note: It is important that you choose a date and time format that accurately reflects the data you are reading from the file. For example, if the file contains date data in the format Month/Day/Year but you choose Day/Month/Year, any date calculations you perform in the dataflow, such as sorting by date, will not reflect the correct date. In addition, records may fail type conversion, in which case the failure behavior specified in the type conversion options in Spectrum Management Console or Spectrum Enterprise Designer will take effect.

- a) In the **Locale** field, select the country whose formatting convention you want to use. Your selection will determine the default values in the **Format** field. For date data, 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 **Format** field, select the format for the data. The format depends on the data type of the field. A list of the most commonly used formats for the selected locale is provided.

An example of the selected format is displayed to the right of the **Format** field.

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 32. To specify

your own number format, type the format into the file using the notation described in **Number Patterns** on page 34.

- In the **Start position** field, enter the position of the first character of the field, and in the **Length** field enter the number of characters in the field.

For example, if the field starts at the tenth character of the record and is five characters long, you would specify a starting position of 10 and a length of 5.

- Click **Add**.
- Repeat this process to add additional fields to the record type, or click **Close** if you are done adding fields.

Flattening Variable Format Data

Variable format file data often contains records that have a hierarchical relationship, with one record type being a parent to other record types. Since many stages require data to be in a flat format, so you may have to flatten the data in order to make the data usable by downstream stages. For example, consider this input data:

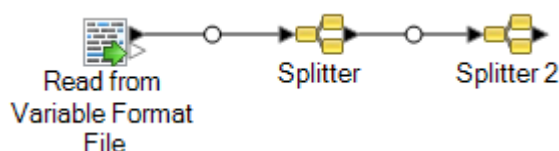
```
001 Joe,Smith,M,100 Main St,555-234-1290
100 CHK12904567,12/2/2007,6/1/2012,CHK
200 1000567,1/5/2012,Fashion Shoes,323.12
001 Anne,Johnson,F,1202 Lake St,555-222-4932
100 CHK238193875,1/21/2001,4/12/2012,CHK
200 1000232,3/5/2012,Blue Goose Grocery,132.11
200 1000232,3/8/2012,Trailway Bikes,540.00
```

You may want to flatten the records so that you have one record per transaction. In the above example, that would mean taking the transaction records (records with the tag 200) and flattening them to include the account owner information (records with the tag 001) and the account details (records with the tag 100).

The following procedure describes how to use Splitter stages to flatten records.

- Add a Read from Variable Format File stage to your data flow and configure the stage. For more information, see **Read from Variable Format File** on page 352.
- Add a Splitter stage and connect it to Read from Variable Format File.
- Add additional Splitter stages as needed so that you have one splitter stage for each child record type in your input data.
- Connect all the Splitter stages.

You should now have a data flow that looks like this:



5. Double-click the first Splitter stage to open the stage options.
6. In the **Split at** field, select one of the child record types.
7. Click **OK**.
8. Configure each additional Splitter stage, selecting a different child record type in each Splitter's **Split at** field.

Read From XML

The Read from XML stage reads an XML file into a job or subflow. It defines the file's path and data format, including XML schema and data element details.

Simple XML elements are converted to flat fields and passed on to the next stage. Simple XML data consists of records made up of XML elements that contain only data and no child elements. For example, this is a simple XML data file:

```
<customers>
  <customer>
    <name>Sam</name>
    <gender>M</gender>
    <age>43</age>
    <country>United States</country>
  </customer>
  <customer>
    <name>Jeff</name>
    <gender>M</gender>
    <age>32</age>
    <country>Canada</country>
  </customer>
  <customer>
    <name>Mary</name>
    <gender>F</gender>
    <age>61</age>
    <country>Australia</country>
  </customer>
</customers>
```

Notice that in this example each record contains simple XML elements such as `<name>`, `<gender>`, `<age>`, and `<country>`. None of the elements contain child elements.

The Read from XML stage automatically flattens simple data like this because most stages require data to be in a flat format. If you want to preserve the hierarchical structure, use an **Aggregator** stage after **Read from XML** to convert the data to hierarchical data.

Complex XML elements remain in hierarchical format and are passed on as a list field. Since many stages require data to be in a flat format, so you may have to flatten complex XML to make the data usable by downstream stages. See [Flattening Complex XML Elements](#) on page 371 for more information.

Note: Read From XML does not support the XML types `xs:anyType` and `xs:anySimpleType`.

*File Properties Tab***Table 30: File Properties Tab**

Option Name	Description
Schema file	<p>Specifies the path to an XSD schema file. Click the ellipses button (...) to locate the file you want. Note that the schema file must be on the server in order for the data file to be validated against the schema. If the schema file is not on the server, validation is disabled.</p> <p>Alternatively, you can specify an XML file instead of an XSD file. If you specify an XML file the schema will be inferred based on the structure of the XML file. Using an XML file instead of an XSD file has some limitations:</p> <ul style="list-style-type: none"> • The XML file cannot be larger than 1 MB. If the XML file is more than 1 MB in size, try removing some of the data while maintaining the structure of the XML. • The data file will not be validated against the inferred schema. <p>Note: If the Spectrum Technology Platform server is running on Linux, remember that file names and paths on these platforms are case sensitive.</p>
Data file	<p>Specifies the path to the XML data file. Click the ellipses button (...) to locate the file you want.</p> <p>Note: If the Spectrum Technology Platform server is running on Linux, remember that file names and paths on these platforms are case sensitive.</p>
Preview	<p>Displays a preview of the schema or XML file. When you specify an XSD file, the tree structure reflects the selected XSD. Once you specify both a schema file and a data file, you can click on the schema elements in bold to see a preview of the data that the element contains.</p>

*Fields Tab***Table 31: Fields Tab**

Option Name	Description
Filter	Filters the list of elements and attributes to make it easier to browse. The filter does not have any impact on which fields are included in the output. It only filters the list of elements and attributes to make it easier to browse.
XPath	The XPath column displays the XPath expression for the element or attribute. It is displayed for information purposes only. For more information about XPath, review this page .
Field	The name that will be used in the dataflow for the element or attribute. To change the field name, double-click and type the field name you want.

Option Name	Description
-------------	-------------

Type

Option Name

Description

The data type to use for the field.

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. Dates must be in the format *yyyy-MM-dd*. For example, 2012-01-30.

datetime A data type that contains a month, day, year, and hours, minutes, and seconds. Datetime must be in the format *yyyy-MM-dd'T'HH:mm:ss*. For example, 2012-01-30T06:15:30

double A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. 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 $(2-2^{-23}) \times 2^{127}$. 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).

list Strictly speaking, a list is not a data type. However, when a field contains hierarchical data, it is treated as a "list" field. In Spectrum Technology Platform a list is a collection of data consisting of multiple values. For example, a field Names may contain a list of name values. This may be represented in an XML structure as:

```
<Names>
  <Name>John Smith</Name>
  <Name>Ann Fowler</Name>
</Names>
```

It is important to note that the Spectrum Technology Platform list data type different from the XML schema list data type in that the XML list data type is a simple data type consisting of multiple values, whereas the Spectrum Technology Platform list data type is similar to an XML complex data type.

long A numeric data type that contains both negative and positive whole numbers between -2^{63} (-9223372036854775808) and $2^{63}-1$ (9223372036854775807).

string A sequence of characters.

time A data type that contains the time of day. Time must be in the format

Option Name	Description
	<i>HH:mm:ss</i> . For example, 21:15:59.
Include	Specifies whether to make this field available in the dataflow or to exclude it.

Example: Simple XML File

In this example, you want to read this file into a dataflow:

```
<addresses>
  <address>
    <addressline1>One Global View</addressline1>
    <city>Troy</city>
    <state>NY</state>
    <postalcode>12128</postalcode>
  </address>
  <address>
    <addressline1>1825B Kramer Lane</addressline1>
    <city>Austin</city>
    <state>TX</state>
    <postalcode>78758</postalcode>
  </address>
</addresses>
```

In this example, you could choose to include the `<addressline1>`, `<city>`, `<state>`, and `<postalcode>`. This would result in one record being created for each `<address>` element because `<address>` is the common parent element for `<addressline1>`, `<city>`, `<state>`, and `<postalcode>`.

Flattening Complex XML Elements

Most stages in a dataflow require data to be in a flat format. This means that when you read hierarchical data from an XML file into a dataflow, you will have to flatten it if the data contains complex XML elements. A complex XML element is an element that contains other elements or attributes. For example, in the data file the `<address>` element and the `<account>` element are complex XML elements:

```
<customers>
  <customer>
    <name>Sam</name>
    <gender>M</gender>
    <age>43</age>
    <country>United States</country>
    <address>
```

```

        <addressline1>1253 Summer St.</addressline1>
        <city>Boston</city>
        <stateprovince>MA</stateprovince>
        <postalcode>02110</postalcode>
    </address>
    <account>
        <type>Savings</type>
        <number>019922</number>
    </account>
</customer>
<customer>
    <name>Jeff</name>
    <gender>M</gender>
    <age>32</age>
    <country>Canada</country>
    <address>
        <addressline1>26 Wellington St.</addressline1>
        <city>Toronto</city>
        <stateprovince>ON</stateprovince>
        <postalcode>M5E 1S2</postalcode>
    </address>
    <account>
        <type>Checking</type>
        <number>238832</number>
    </account>
</customer>
<customer>
    <name>Mary</name>
    <gender>F</gender>
    <age>61</age>
    <country>Australia</country>
    <address>
        <addressline1>Level 7, 1 Elizabeth Plaza</addressline1>
        <city>North Sydney</city>
        <stateprovince>NSW</stateprovince>
        <postalcode>2060</postalcode>
    </address>
    <account>
        <type>Savings</type>
        <number>839938</number>
    </account>
</customer>
</customers>

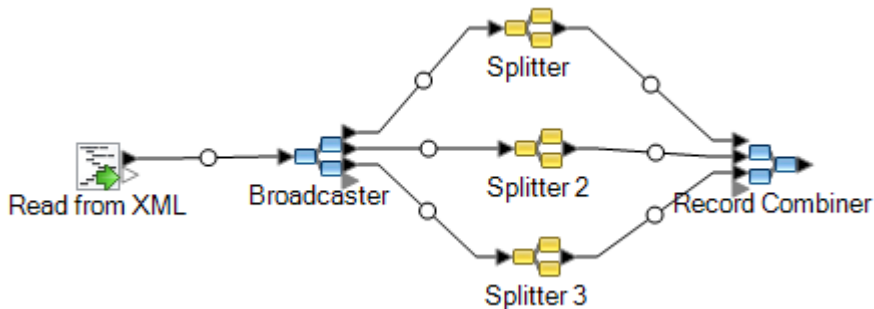
```

This procedure describes how to use Splitter stages to flatten XML data containing multiple complex XML elements.

Note: If your data contains a single complex XML element, you can use a single Splitter stage to flatten the data by simply connecting the Read from XML stage to the Splitter stage. You do not need to use the Broadcaster and Record Combiner stages as described in this procedure for data files containing a single complex XML element.

1. Add a Read from XML stage to your data flow and configure the stage. For more information, see [Read From XML](#).
2. Add a Broadcaster stage and connect Read from XML to it.
3. Add a Splitter stage for each complex XML element in your data.
4. Connect the Broadcaster stage to each Splitter.
5. Add a Record Combiner stage and connect each Splitter to it.

You should now have a data flow that looks like this:



6. Double-click the first Splitter stage to open the stage options.
7. In the **Split at** field, select one of the complex fields. In the example data file above, this could be the address field.
8. Click **OK**.
9. Configure each additional Splitter stage, selecting a different complex XML element in each Splitter's **Split at** field.

The data flow is now configured to take XML input containing records with complex XML elements and flatten the data. The resulting records from Record Combiner can be sent to any stage that requires flat data. For example, you could attached the Record Combiner stage to a Validate Address stage for address validation.

SQL Command

SQL Command executes one or more SQL commands for each record in the data flow. You can use SQL Command to:

- Execute complex INSERT/UPDATE statements, such as statements that have subqueries/joins with other tables.
- Update tables after inserting/updating data to maintain referential integrity.
- Update or delete a record in a database before a replacement record is loaded.
- Update multiple tables in a single transaction.

You can execute additional SQL commands before and after executing the main SQL commands, and you can invoke stored procedures.

Note: To execute a stored procedure, use this syntax:

Call *<Procedure Name>*

Stored procedures invoked from SQL Command must not use OUT parameters.

Note: Significant performance improvements can be achieved by using multiple runtime instances of SQL Command. To specify multiple runtime instances, click the **Runtime** button.

General

The **General** tab is where you specify dynamic SQL statements that you want to execute once for each record. The following table lists the options available on the **General** tab.

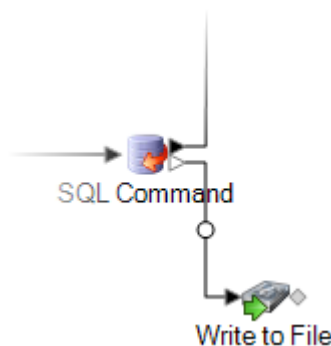
Option	Description
Connection	<p>Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Spectrum Management Console. If you need to make a new database connection, or modify or delete an existing database connection, click Manage Connections.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection name Enter a name for the connection. The name can be anything you choose.</p> <p>Database driver Select the appropriate database type.</p> <p>Connection options Specify the host, port, instance, user name, and password to use to connect to the database.</p>
SQL statements	<p>Enter the SQL statements you want to run for each record in the dataflow. As you begin to type, an auto-complete pop-up window will display the valid SQL commands. Separate multiple SQL statements with a semicolon (;).</p> <p>To specify a value from a dataflow field, use this syntax:</p> <pre> \${<field name>} </pre> <p>Where <field name> is the name of a field in the data flow.</p> <p>For example,</p> <pre> UPDATE MyDatabase.dbo.customer SET name=\${Name} WHERE id=\${ID}; </pre> <p>In this example <code>\${Name}</code> will be replaced with the value from the dataflow's Name field and <code>\${ID}</code> will be replaced with the value from the dataflow's ID field.</p> <p>Note: Queries must use the fully-qualified name. For example, <code>MyDatabase.dbo.customer</code>.</p>

Option	Description
Transaction processing	<p>Specifies whether to process records in batches or to process all records at the same time. One of these:</p> <p>Batch size Groups records into batches of the size you specify and processes one batch at a time.</p> <p>Entire Run Creates one large batch for all records and processes all transactions at the same time.</p>

Error processing	<p>Specifies what to do if an error is encountered while executing the SQL commands. One of the following:</p> <p>Do not terminate the data flow on error The data flow continues to run if the database returns an error while executing the SQL commands.</p> <p>Terminate the data flow after encountering this many errors The data flow will stop running after the database returns the specified number of errors.</p>
------------------	---

Note: If there is a syntax error in the SQL, the data flow will always terminate regardless of which setting you choose here.

In addition, you can optionally write error records to a sink by connecting the SQL Command error port to the type of sink you want. The error port is the white triangle on the right side of the stage icon in the data flow. For example, to write error records to a flat file, you would connect the SQL Command error port to a Write to File stage, as shown here:



Pre/Post SQL

The **Pre/Post SQL** tab is where you specify SQL statements that you want to execute once per data flow run, as opposed to once per record as is the case with the SQL you specify on the **General** tab. The following table lists the options available on the **Pre/Post SQL** tab.

Option	Description
Pre-SQL	<p>Type one or more SQL statements that you want to execute before the records coming into the stage are processed. The SQL statements you enter here are executed once per run after the data flow starts running but before the SQL Command stage processes the first records.</p> <p>An example use of pre-SQL would be to create a table for the records that will be processed.</p>
Autocommit pre-SQL	<p>Check this box to commit the pre-SQL statements before executing the SQL statements on the General tab.</p> <p>If you do not check this box, the pre-SQL statements will be committed in the same transaction as the SQL statements on the General tab.</p> <p>Note: If you check neither the Autocommit pre-SQL nor the Autocommit post-SQL boxes, then all SQL statements for the stage are committed in one transaction.</p>
Post-SQL	<p>Type one or more SQL statements that you want to execute after all the records are processed. The SQL statements you enter here are executed once per run after the SQL Command stage is finished but before the data flow finishes.</p> <p>An example use of pre-SQL would be to build an index after processing the records.</p>
Autocommit post-SQL	<p>Check this box to commit the post-SQL statements in their own transaction after the SQL commands on the General tab are committed.</p> <p>If you do not check this box, the post-SQL statements will be committed in the same transaction as the SQL statements on the General tab.</p> <p>Note: If you check neither the Autocommit pre-SQL nor the Autocommit post-SQL boxes, then all SQL statements for the stage are committed in one transaction.</p>

Runtime Tab

The **Runtime** tab displays **Stage Options** and gives you the flexibility of defining default values for the stage options.

Field Name	Description
Stage Options	<p>This section lists the dataflow options used in the SQL query of this stage and allows you to provide a default value for all these options. The Name column lists the options while you can enter the default values in the corresponding Value column.</p> <p>Note: The default value provided here is also displayed in the Map dataflow options to stages section of the Dataflow Options dialog box. The dialogue box also allows you to change the default value. In case of a clash of default values provided for an option through Stage Options, Dataflow Options, and Job Executor the order of precedence is: Value provided through Job Executor > Value defined through the Dataflow Options dialogue box > Value entered through the Stage Options.</p>

Specifying SQL Command at Runtime

This procedure describes how to configure a dataflow to support runtime options for SQL Command and also how to specify the job executor arguments to do this.

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. Expand the SQL Command stage.
6. Select a SQL Command option. It can be **PreSqlCommand**, **SqlCommand**, or **PostSqlCommand**.

PreSqlCommand SQL statements that you want to execute before the records coming into the stage are processed. These SQL statements are executed once per run after the dataflow starts running but before the SQL Command stage processes the first record.

An example use of pre-SQL would be to create a table for the records that will be processed.

SqlCommand SQL statements you want to execute for each record in the dataflow.

PostSqlCommand SQL statements that you want to execute after all the records are processed. These SQL statements are executed once per run after the SQL Command stage is finished but before the dataflow finishes.

An example use of post-SQL would be to build an index after processing the records.

The selected SQL Command option name is displayed in **Option name** and **Option label** fields. This is the option name that will have to be specified at run time in order to set this option.

7. Enter a description of the option in the **Description** field.
8. In the **Target** field, select the option **Selected stage(s)**.
9. 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.
10. 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 845.

11. Click **OK**.
12. Continue adding options as desired.
13. Click **OK** in the **Dataflow Options** dialog box when you are done adding options.
14. Save and expose the dataflow.
15. Create a text file containing the SQL statement you want to use at runtime.

The text file may look like this:

```
SqlCommand = UPDATE CustomersSET
ContactName='Alfred Schmidt'
City='Hamburg'
WHERE CustomerName='Alfreds Futterkiste';
```

In this example, SqlCommand is one of the SQL Command stage's option names.

16. Use the `-o` argument when running a job executor from command line.

```
java -jar jobexecutor.jar -h "noipa019sh-11" -u "admin" -p "admin" -s
"8080" -o "options.txt" -j "FetchOracleData" -w
```

The filename (options.txt) specifies a name of the text file that you created in step 14.

For more information, see [Running A Job from the Command Line](#) on page 378

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 <i>delimiter</i>	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 <i>property file</i>	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 828.
No	-h <i>host name</i>	Specifies the name or IP address of the Spectrum Technology Platform server.
No	-i <i>poll interval</i>	Specifies how often to check for completed jobs, in seconds. This applies only in synchronous mode.
Yes	-j <i>job name</i>	A comma-separated list of jobs to run. Job names are case-sensitive. Jobs are started in the order listed.
No	-n <i>email list</i>	Specifies a comma-separated list of additional email addresses for configured job notifications.
No	-o <i>property file</i>	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 843. For example, a flow options properties file for a flow that contains an Assign GeoTAX Info stage may look like this: <pre>OutputCasing=U UseStreetLevelMatching=N TaxKey=T Database.GTX=gsl</pre>
Yes	-p <i>password</i>	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: <ul style="list-style-type: none"> • 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

Required	Argument	Description
		<ul style="list-style-type: none"> • Position 8 - Number of malformed records • Position 9 - Currently unused <p>For example:</p> <pre>MySimpleJob 4 succeeded 04/09/2019 14:50:47 04/09/2019 14:50:47 100 0 0 </pre> <p>The information is delimited using the delimiter specified in the <code>-d</code> argument.</p>
No	<code>-s port</code>	The socket (port) on which the Spectrum Technology Platform server is running. The default is 8080.
No	<code>-t timeout</code>	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	<code>-u user name</code>	The login name of the user.
No	<code>-v</code>	Return verbose output.
No	<code>-w</code>	Runs job executor in synchronous mode. This means that job executor remains running until the job completes. If you do not specify <code>-w</code> , 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	See Override File Location	Overrides the input or output file specified in Read from File or Write to File. For more information, see Overriding Job File Locations on page 823.
No	See Override File Format	Overrides the file layout definition specified in Read from File or Write to File with one defined in a schema file. For more information, see Overriding the File Format at the Command Line on page 825.

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."

Executing SQL Commands Before or After a Dataflow

The **Execute SQL** activity performs operations on database at any point during a process flow. This activity allows you to run the SQL statements both before and after the execution of Spectrum Technology Platform dataflow or an external program. For example, the **Execute SQL** activity can be used to delete indexes before the execution of a Spectrum Technology Platform dataflow and to create indexes again after the execution of the dataflow. To execute SQL statements using **Execute SQL** activity, you must create a process flow.

Note: Please refer to *Spectrum Dataflow Designer Guide* for instructions on how to create and schedule a process flow.

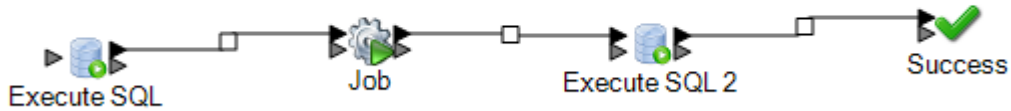
1. Drag the **Execute SQL** activity to the canvas.
2. Double click the **Execute SQL** activity.
3. Select a database connection you want to use.

If you need to make a new database connection, or modify or delete an existing database connection, click **Manage**.

If you are adding or modifying a database connection, complete these fields:

Connection name	Enter a name for the connection. The name can be anything you choose.
Database driver	Select the appropriate database type.
Connection options	Specify the host, port, instance, user name, and password to use to connect to the database.

4. Write the SQL statement in the **SQL statement(s)** box.
By default, the **Terminate flow on error** option is checked which means that the process flow will be terminated if an exception occurs. If the option **Terminate flow on error** is unchecked and an exception occurs, the process flow will not stop and the exception will be logged in the server logs.
5. Add the action you want a process flow to perform.
You can add a job by dragging a job's icon to the canvas, or add an external program by dragging a Run Program icon onto the canvas.
6. Connect the two activities.
7. Add additional **Execute SQL** activity as needed.
Refer step 2 to step 5 for performing actions on **Execute SQL**.
8. When you have added all the jobs, Run Program and Execute SQL activities you want to execute in the process flow, drag a Success activity onto the canvas and connect it to the last activity in the process flow.



9. Run the Process flow.

Transposer

Transposer converts columns to rows. Transposing data is the opposite of pivoting data using the Group Statistics stage, which transforms row data into columns.

To understand Transposer, consider the following example. A table contains four quarters of sales data and you want to add all the revenue generated and to analyze the growth achieved in first three quarters. To accomplish this, use Transposer to create a column containing all the revenue of three transposed quarters. Using Transposer to add all the revenues generated in different columns into a column potentially improves performance instead of adding them in different columns.

The following table explains the options in the Transposer dialog box.

Option	Description
Transposed fields header	Type a header name for the column that will contain those columns which are to be transposed. This new column is automatically added to the dataflow.
Transposed values header	Type a header name for the column that will contain the transposed column values. This new column is automatically added to the dataflow.
Retain transposed fields	Check this option to retain all the transposed fields as columns in the output.
Field Name	Displays all the column headers of input file.

Option Description

Type

Displays the data type of the respective fields (column headers).

The columns to be transposed should have compatible data type in the input source file. Below is the compatibility matrix. The tick marked grids correspond to the compatible data types.

	Integer	Long	String	Date/Time	Double	Big Decimal	Time	Date
Integer	✓	✓	✓		✓	✓		
Long	✓	✓	✓		✓	✓		
String	✓	✓	✓	✓	✓	✓	✓	✓
Datetime			✓	✓				
Double	✓	✓	✓		✓			
Bigdecimal	✓	✓	✓			✓		
Time			✓				✓	
Date			✓					✓

Transposed

Check the box next to each field that you want to convert to a column. In order to prevent a column from getting transposed and retain it in the output, clear the check box.

Example Use of Transposer

The following input data contains four quarters of sales by store. Note that Q1, Q2, Q3, and Q4 represent four quarters of sales (in millions).

Store (US)	Q1	Q2	Q3	Q4
New York	100.00	200.10	300.00	400.00
California	250.10	450.00	550.00	650.00

Store (US)	Q1	Q2	Q3	Q4
Illinois	150.00	250.10	350.00	450.00

The cases mentioned below illustrate the behavior of Transposer using the options provided in the stage. Note that Quarter is the column name for Transposed fields header and Revenue is the column name for Transposed fields values.

Case 1

Suppose you want columns Q1, Q2, and Q3 to be transposed and Q4 to be retained in the output. To do this, check the box under the **Transposed** header next to each column which is to be transposed. You will now see Q1, Q2, and Q3 as rows whereas Q4 will be retained as a column in the output.

Store (US)	Quarter	Revenue	Q4
New York	Q1	100.00	400.00
New York	Q2	200.10	400.00
New York	Q3	300.00	400.00
California	Q1	250.10	650.00
California	Q2	450.00	650.00
California	Q3	550.00	650.00
Illinois	Q1	150.00	450.00
Illinois	Q2	250.10	450.00
Illinois	Q3	350.00	450.00

Case 2

Suppose you want columns Q1 and Q2 to be transposed and Q3 and Q4 to be retained in the output. In addition, you also want to retain all the transposed fields (Q1 and Q2) as columns in the output. To do this, check the option **Retain**

transposed fields and the box under the **Transposed** header next to each column to be transposed. You will now see Q1 and Q2 as rows whereas Q3 and Q4 will be retained as columns in the output along with Q1 and Q2.

Store (US)	Quarter	Revenue	Q1	Q2	Q3	Q4
New York	Q1	100.00	100.00	200.10	300.00	400.00
New York	Q2	200.10	100.00	200.10	300.00	400.00
California	Q1	250.10	250.10	450.00	550.00	650.00
California	Q2	450.00	250.10	450.00	550.00	650.00
Illinois	Q1	150.00	150.00	250.10	350.00	450.00
Illinois	Q2	250.10	150.00	250.10	350.00	450.00

Unique ID Generator

The Unique ID Generator stage creates a unique key that identifies a specific record. A unique ID is crucial for data warehouse initiatives in which transactions may not carry all name and address data, but must be attributed to the same record or contact. A unique ID may be implemented at the individual, household, business, and premises level. Unique ID Generator provides a variety of algorithms to create unique IDs.

The unique ID is based on either a sequential number or date and time stamp. In addition, you can optionally use a variety of algorithms to generate data to appended to the ID, thereby increasing the likelihood that the ID will be unique. The sequential number or date and time stamp IDs are required and cannot be removed from the generated ID.

Unique ID Generator can be used to generate a non-unique key using one of the key generation algorithms. In non-unique mode, you can create keys to use for matching. This may be useful in a data warehouse where you have already added keys to a dimension and you want to generate a key for new records in order to see if the new records match an existing record.

This example shows that each record in the input is assigned a sequential record ID in the output.

Record	RecordID
John Smith	0
Mary Smith	1
Jane Doe	2
John Doe	3

The Unique ID stage produces a field named RecordID which contains the unique ID. You can rename the RecordID field as required.

Defining a Unique ID

By default, the Unique ID Generator stage creates a sequential ID, with the first record having an ID of 0, the second record having an ID of 1, the third record having an ID of 2, and so forth. If you want to change how the unique ID is generated, follow this procedure.

1. In the Unique ID Generator stage, on the **Rules** tab, click **Modify**.
2. Choose the method you want to use to generate the unique ID.
For more information, see [Unique ID Definition Methods](#) on page 91.
3. Click **OK**.

Unique ID Definition Methods

Options	Description
Sequential Numeric tag starting at	<p>Assigns an incremental numeric value to each record starting with the number you specify. If you specify 0, the first record will have an ID of 0, the second record will have an ID of 1, and so on.</p> <p>Note: For this Unique Key, ensure you do not increase the Runtime instances (in the Runtime Performance tab) beyond 1 as this can create duplicate IDs.</p>

Options	Description
Sequential Numeric tag starting at value in a database field	<p>Assigns an incremental numerical value to each record starting with the maximum number read from the database field. This number is then incremented by 1 and assigned to the first record. For example, if the number read from the database field is 30, the first record will have an ID of 31, the second record will have an ID of 32, and so forth.</p> <p>Connection Select the database connection you want to use. Your choices vary depending on what connections are defined in the Connection Manager of Spectrum Management Console. If you need to make a new database, or modify or delete an existing connection, click Manage.</p> <p>If you are adding or modifying a database connection, complete these fields:</p> <p>Connection name Enter a name for the connection. This can be anything you choose.</p> <p>Database driver Select the appropriate database type.</p> <p>Connection options Specify the host, port, instance, user name, and password to use to connect to the database.</p> <p>Table view Specifies the table or view in the database that you want to query.</p> <p>Database field Select a column from the list to generate a unique key.</p> <p>The supported data types for unique ID generation are:</p> <p>long A numeric data type that contains both negative and positive whole numbers between -2^{63} ($-9,223,372,036,854,775,808$) and $2^{63}-1$ ($9,223,372,036,854,775,807$).</p> <p>integer A numeric data type that contains both negative and positive whole numbers between -2^{31} ($-2,147,483,648$) and $2^{31}-1$ ($2,147,483,647$).</p> <p>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.</p> <p>double A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. In E notation, the range of values is $-1.79769313486232E+308$ to $1.79769313486232E+308$.</p> <p>float A numeric data type that contains both negative and positive single precision numbers between 2^{-149} and $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is $-3.402823E+38$ to $3.402823E+38$.</p>

Options	Description
Date/Time stamp	Creates a unique key based on the date and time stamp instead of sequential numbering.
UUID	Creates a universally unique 32-digit identifier key for each record. The digits in the key are displayed in five groups separated by hyphens, in the form 8-4-4-4-12 for a total of 36 characters (32 alphanumeric characters and four hyphens). Example: 123e4567-e89b-12d3-a456-432255330000
Off	Select this option only if you want to generate a non-unique key using an algorithm.

Using Algorithms to Augment a Unique ID

Unique ID Generator generates a unique ID for each record by either numbering each record sequentially or generating a date-time stamp for each record. You can optionally use algorithms to append additional information to the sequential or date-time unique ID, thereby creating a more complex unique ID and one that is more likely to be truly unique.

1. In the Unique ID Generator stage, click **Add**.
2. In the **Algorithm** field, select the algorithm you want to use to generate additional information in the ID.

Consonant	Returns specified fields with consonants removed.
Double Metaphone	Returns a code based on a phonetic representation of their characters. Double Metaphone is an improved version of the Metaphone algorithm, and attempts to account for the many irregularities found in different languages.
Koeln	Indexes names by sound as they are pronounced in German. Allows names with the same pronunciation to be encoded to the same representation so that they can be matched, despite minor differences in spelling. The result is always a sequence of numbers; special characters and white spaces are ignored. This option was developed to respond to limitations of Soundex.
MD5	A message digest algorithm that produces a 128-bit hash value. This algorithm is commonly used to check data integrity.
Metaphone	Returns a Metaphone coded key of selected fields. Metaphone is an algorithm for coding words using their English pronunciation.
SpanishMetaphone	Returns a Metaphone coded key of selected fields for the Spanish language. This metaphone algorithm codes words using their Spanish pronunciation.

Metaphone 3 Improves upon the Metaphone and Double Metaphone algorithms with more exact consonant and internal vowel settings that allow you to produce words or names more or less closely matched to search terms on a phonetic basis. Metaphone 3 increases the accuracy of phonetic encoding to 98%. This option was developed to respond to limitations of Soundex.

3. In the **Field name** field, choose the field to which you want to apply the algorithm. For example, if you chose the soundex algorithm and chose a field named City, the ID would be generated by applying the soundex algorithm to the data in the City field.
4. If you selected the substring algorithm, specify the portion of the field you want to use in the substring:
 - a) In the **Start position** field, specify the position in the field where you want the substring to begin.
 - b) In the **Length** field, select the number of characters from the start position that you want to include in the substring.

For example, say you have this data in a field named LastName:

Augustine

If you specified 3 as the start position and 6 as the end position, the substring would produce:

gustin

5. Check the **Remove noise characters** box to remove all non-numeric and non-alpha characters such as hyphens, white space, and other special characters from the field before applying the algorithm.
6. For consonant and substring algorithms, you can sort the data in the field before applying the algorithm by checking the **Sort input** box. You can then choose to sort either the characters in the field or terms in the field in alphabetical order.
7. Click **OK** to save your settings.
8. Repeat as needed if you want to add additional algorithms to produce a more complex ID.

Note: The unique key definition is always displayed in a different color and cannot be deleted.

Defining a Non-Unique ID

Unique ID Generator can be used to generate a non-unique key using one of the key generation algorithms. In non-unique mode, you can create keys to use for matching. This may be useful in a data warehouse where you have already added keys to a dimension and you want to generate a key for new records in order to see if the new records match an existing record.

1. In the Unique ID Generator stage, on the **Rules** tab, click **Modify**.
2. Select **Off**.

This turns off the unique ID portion of the ID generation rules. With this option off, only the algorithm you choose in the steps below are used to create the ID. This means that any records that have the same data in the fields you use to generate the ID will have the same ID. You can then use the ID for matching.

3. Click **OK**.
4. At the warning prompt, click **Yes**.
5. In the Unique ID Generator stage, click **Add**.
6. In the **Algorithm** field, select the algorithm you want to use to generate additional information in the ID.

Consonant	Returns specified fields with consonants removed.
Double Metaphone	Returns a code based on a phonetic representation of their characters. Double Metaphone is an improved version of the Metaphone algorithm, and attempts to account for the many irregularities found in different languages.
Koeln	Indexes names by sound as they are pronounced in German. Allows names with the same pronunciation to be encoded to the same representation so that they can be matched, despite minor differences in spelling. The result is always a sequence of numbers; special characters and white spaces are ignored. This option was developed to respond to limitations of Soundex.
MD5	A message digest algorithm that produces a 128-bit hash value. This algorithm is commonly used to check data integrity.
Metaphone	Returns a Metaphone coded key of selected fields. Metaphone is an algorithm for coding words using their English pronunciation.
SpanishMetaphone	Returns a Metaphone coded key of selected fields for the Spanish language. This metaphone algorithm codes words using their Spanish pronunciation.
Metaphone 3	Improves upon the Metaphone and Double Metaphone algorithms with more exact consonant and internal vowel settings that allow you to produce words or names more or less closely matched to search terms on a phonetic basis. Metaphone 3 increases the accuracy of phonetic encoding to 98%. This option was developed to respond to limitations of Soundex.

7. In the **Field name** field, choose the field to which you want to apply the algorithm. For example, if you chose the soundex algorithm and chose a field named City, the ID would be generated by applying the soundex algorithm to the data in the City field.
8. If you selected the substring algorithm, specify the portion of the field you want to use in the substring:
 - a) In the **Start position** field, specify the position in the field where you want the substring to begin.

- b) In the **Length** field, select the number of characters from the start position that you want to include in the substring.

For example, say you have this data in a field named LastName:

Augustine

If you specified 3 as the start position and 6 as the end position, the substring would produce:

gustin

9. Check the **Remove noise characters** box to remove all non-numeric and non-alpha characters such as hyphens, white space, and other special characters from the field before applying the algorithm.
10. For consonant and substring algorithms, you can sort the data in the field before applying the algorithm by checking the **Sort input** box. You can then choose to sort either the characters in the field or terms in the field in alphabetical order.
11. Click **OK** to save your settings.
12. Repeat as needed if you want to add additional algorithms to produce a more complex ID.

Note: The unique key definition is always displayed in a different color and cannot be deleted.

Write to Cache

Write to Cache loads output from a dataflow into a global cache, making the data available for lookup from the Query Cache stage. Using a global cache for data lookups improves performance compared to lookups to databases.

A global cache is system-wide, shared cache that will reside in memory. Choose a global cache if you want the cache to be available to multiple dataflows or when data does not change often or remains relatively static and when storage is not limited. A global cache is static as you can write to it only once. The cache can not be updated once it has been created.

The cache size is set to 500K records (default). You can configure it by using the following property of **jmx-console**:

MBean:

```
com.pb.spectrum.edi.managers.config.impl:manager=EDIGlobalCacheConfigManager
```

Description: Data Global Cache Configuration Manager

```
MaxCacheSize = 500000
```

Note: Write to Cache overwrites the cache each time the dataflow runs.

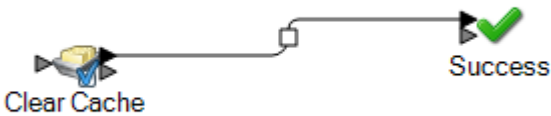
General

Option Name	Description
Cache name	Specifies the name you want to give to the cache. If there are caches already on the system, they are listed and you can choose one if you want to populate the existing cache with new data. To create a new cache, type the name you want for the new cache. The name must begin with a letter. It can contain an underscore but no other special characters. The name can contain numeric values.
Cache Fields	This column lists the field names that will be used in the cache. If you wish to change a field name, click the field name and enter a new name.
Stage Fields	This column lists the field names used in the dataflow. You cannot modify these field names.
Type	This column lists the data type of each field.
Include	Check the box in this column to have the field written to the cache. Clear the box if you do not want the field written to the cache.
Key Field	<p>Check the box in this column if you want the field to be used as a key in the Query Cache stage. For example, if you have a dataflow field named AccountNumber and you want the Query Cache stage to look up data by querying for a matching value in the AccountNumber field, you would check the box in the Key Field column for the AccountNumber field.</p> <p>The fields you specify as key fields are available for selection in the Query Cache stage as key fields.</p>

Clearing a Global Cache

To clear a global cache you must create and execute a process flow. The process flow must contain a Clear Cache activity. The Clear Cache activity clears the global cache but does not delete it. You can also clear the cache automatically by scheduling a process flow.

Note: Please see *Dataflow Designer Guide* for instructions on how to create and schedule a process flow.



To manually clear the global cache data, follow these steps:

1. Drag the **Clear Cache** activity to the canvas.
2. Drag the **Success** activity to the canvas.
3. Connect the two activities.
4. Double-click the **Clear Cache** activity.
5. Select the cache. You can also select multiple caches to clear their data.
The caches that you create in Write to Cache stage are listed in Clear Cache activity.
6. Run the process flow.

Write to DB

The **Write to DB** stage writes the output of a dataflow to a database. The stage writes all values of the `date` datatype as `String` values. This is the behavior of the *jtDS driver*, which is the default driver used by Spectrum. To handle all `date` datatype values as is, use Microsoft's JDBC driver.

Note: The stage supports reading data from and writing data to HDFS 3.x and Hive 2.1.1. The support includes:

- Connectivity to Hive from Spectrum on Windows
- Support and connectivity to Hive version 2.1.1 from Spectrum with high availability
- Support to Read and Write from Hive DB (JDBC) via Model Store connection


Also see [Best Practices for connecting to HDFS 3.x and Hive 2.1.1](#).

Note: Significant performance improvements can be achieved by using multiple runtime instances of Write to DB. To specify multiple runtime instances, click the **Runtime** button.

Configuring the General tab

1. From the **Connection** drop down list, select the connection for the database you want to use.
2. To make a new database connection, click **Manage**. For more information about creating database connections, see [Database Connection Manager](#) on page 397

Note: This option is available only through the **Spectrum Enterprise Designer**.

3. To select a table or view from the database, click the browse button  and to go to the table or view that you want to use.

When you browse a table and select it, the **Table Schema**, including **Database Fields**, **Stage Fields**, and **Data Types** are displayed. A **Preview** of the table is also available.

Note: If you are writing to a SQL database, you cannot write to views that reference more than one table. This is due to a limitation in SQL Server.

4. To create a new table in the database, click **Create Table**  and in the pop-up that appears, select the **Table Owner** and specify the **Table Name**.

Note: Table names are case sensitive

Note: If you do not have an input stage (such as **Read from File** or **Read from DB**) linked to the **Write to DB** stage, you will get this error message: *Cannot create table without Table schema defined. Please make sure you have upstream fields defined for this stage.*

5. In the Table Schema, specify these details:
 - a. Indicate the primary key by selecting the corresponding **Primary key** check box.
 - b. Select the **Include** check box to specify the fields you want to write to the new table.
 - c. For string data type, specify the fields's length in the **Width** column.

Note: The default is 512.

- d. If **Allow Null** is checked and the **Input Fields** contains a null value, then the dataflow writes the null value to the database.
- e. You can edit the column name by changing the value in the corresponding **Output Fields**.

The **Create Table** button supports table creation in these databases:

- Axion
- DB2
- Derby or Cloudscape
- Firebird
- HSQLDB
- Interbase
- MaxDB or SapDB
- McKoi
- MySQL
- Oracle
- PostgreSQL
- SQL Server
- Sybase

Note: For DB2 databases, if you try to create a table and the page size is smaller than the total length of all string columns, you will get an error that says "Failed to build body from content. Serializable class not available to broker."

6. Click the **OK** button to close the **Create Table** pop-up and return to **Write to DB Options**.
7. In the **Stage Fields** column of the **Table Schema**, you can specify the field name you want to write to the database corresponding to the **Database Field** column.
8. Mark the Include check box to select the fields you want to write.

Note: To prevent poor performance you should have a sorted index or key in the database table.

Configuring the Runtime tab

Option Name	Description
Write Mode	Specifies an action to take when writing to the database:
	<p>Insert Insert new records into the database but do not update existing records. This is the default setting.</p> <p>Update Update existing records in the database but do not insert new records.</p> <p>Note: If you select Update, the primary key column name used in the input table must match the primary key column name in the output table. If you try to update a table where the primary key column name does not match the input, or where the primary key column is not defined, the update will not work.</p>
	<p>Insert if not able to update Insert new records into the database if the record does not exist, otherwise update the existing record.</p>
Batch commit	Select this option to commit changes to the database after a specified number of records are processed. By default this option is not selected, which means that changes are committed after each record is processed. Selecting this option can significantly improve the performance of the Write to DB stage.

Option Name	Description
Batch size	<p>If you enable the Batch commit option, specifies the number of records to commit to the database in each batch. The default is 1,000. For dataflows created in Spectrum Technology Platform 7.0 and earlier, the default is 100.</p> <p>A larger batch size does not always offer better load performance. Consider these factors when choosing a batch size:</p> <ul style="list-style-type: none"> • 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.
Commit at the end	<p>Select this option to ensure that the commit to database operation occurs after all the records are transferred to the database.</p>
Batch count to commit	<p>Specify a value after which the records are to be committed. Records are committed to the database after every (batch count to commit * batch size) number of records are transferred to the database. For example, if Batch size is set as 1000 and Batch count to commit is set as 3, then the commit occurs after every 3000 records are transferred to the database.</p>
Truncate table before inserting data	<p>Select this option if you want to clear all data from the table before writing to the database.</p>
Drop and recreate the table if it already exists	<p>Select this option to delete and recreate the table before writing the dataflow's output to the table. This option is useful if you want the table's schema to match the fields from the dataflow and not contain any extraneous schema information.</p> <p>The table that will be deleted and recreated is the one specified in the Table/View field on the General tab. For example, if you specify the Customers table in the Table/View field, and you select Drop and recreate the table if it already exists, then the Customers table will be deleted from the database, and a new table named Customers will be created with a schema that matches the actual fields written to the table.</p>

Database Connection Manager

The Database Connection Manager allows you to manage registered database connections. To add, modify, delete, and test connections:

1. In the **Write To DB Options** dialog box, click **Manage**.
2. Click **Add**, **Modify**, or **Delete**.
3. If you are adding or modifying a database connection, complete these fields:
 - Connection name — Enter the name of the new connection.
 - Database driver — Select the appropriate database type.
 - Connection Options — Specify all the options, typically host, port, instance, user name, and password.

Note: You can test the connection by clicking **Test**.
4. If you are deleting a database connection, select the connection you want to remove and click **Delete**.
5. Click the **Apply** button when you are done with the configuration.

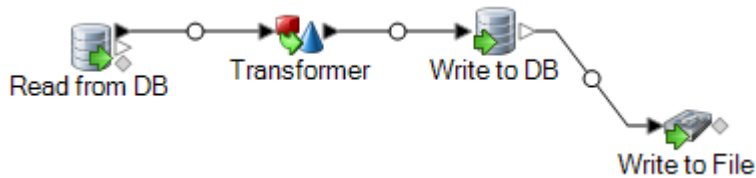
Configuring Error Handling in Write to DB

The Write to DB stage has an error port which allows you to filter out records that cause database errors when writing the record to a database, such as a primary key constraint violation or a unique constraint violation. These records can then be routed along another path in the dataflow while other records are successfully committed. For example, if you are processing 100 records and records 4, 23, and 56 cause a database error, these three records would be routed through the error port while the other 97 records would be committed to the database.

Note: Using the error port is optional. If you do not use the error port, the job will fail if any record causes an error.

1. From the palette, choose the type stage you want to handle error records (for example, Write to File) and drag it onto the canvas. You have a couple options for selecting a stage:
 - To write failed records to a file, drag one of the following onto the canvas: Write to File, Write to XML, or Write to Variable Format File,.
 - To simply discard failed records, drag Write to Null onto the canvas.
2. Connect the error port on Write to DB to the stage you want to handle failed records.

The following example shows the error port on Write to DB connected to a Write to File stage. In this example, records that cause an error when written to the database are instead written to the file specified in the Write to File stage.



When you run the dataflow, records that cause an error are routed through the error port. The records from the error port contain the fields specified in Write to DB plus the following fields:

Error.code	This field contains the numeric error code returned from the database. For example, given the error <code>ORA-00001: unique constraint ANKUSH.SYS_C0010018) violated</code> , the value in the Error.code field would be 1. See your database software's documentation for a listing of error codes.
Error.Message	This field contains the error message returned from the database. For example: <code>ORA-01034 ORACLE not available</code> . In this case, <code>ORACLE not available</code> would be the value in the Error.Message field. See your database software's documentation for a listing of error messages.
Error.SQLState	This field contains the SQLSTATE code which provides detailed information about the cause of the error. For a listing of SQLSTATE codes, see your database software's documentation.
Timestamp	The date and time on the Spectrum Technology Platform server when the error occurred.
Username	The name of the Spectrum Technology Platform user that ran the dataflow.

Write to File

Write to File writes dataflow output to a flat file.

- To write records of varying format, see [Write to Variable Format File](#) on page 430.
- To write records to an XML file, see [Write to XML](#) on page 440.

Tip: You can copy your source and paste it as the sink into your dataflow to quickly set up the file and use the same fields as you defined in your source.

Prerequisite: To write a file to any of the file system connection types, such as FTP, Cloud, Amazon AWS S3, and HDFS, perform these steps:

1. Create a connection to these file servers using **Spectrum Management Console** or **Discovery**. For details, see section [Defining Connections](#).
2. Select the required file path using the **File name** field in **File Properties** tab (described below).

File Properties Tab

Field Name	Description
Server name	Indicates whether the file you select as input is located on the computer running Spectrum Enterprise Designer or on the Spectrum Technology Platform server. If you select a file on the local computer, the server name will be My Computer. If you select a file on the server the server name will be Spectrum Technology Platform.
File name	<p>Specifies the path to the file. Click the ellipses button (...) to locate the file you want.</p> <p>While writing a file to an HDFS file server, these compression formats are supported:</p> <ol style="list-style-type: none"> 1. GZIP (.gz) 2. BZIP2 (.bz2) <p>Note: Include the appropriate extension in the file name, to indicate the desired compression format to be used while writing the file.</p> <p>Attention: If the Spectrum Technology Platform server is running on Linux, remember that file names and paths on these platforms are case sensitive.</p>
Record type	<p>The format of the records in the file. Select one of:</p> <p>Line Sequential A text file in which records are separated by an end-of-line (EOL) character such as a carriage return or line feed (CR or LF) and each field has a fixed starting and ending character position.</p> <p>Fixed Width A text file in which each record is a specific number of characters in length and each field has a fixed starting and ending character position.</p> <p>Delimited A text file in which records are separated by an end-of-line (EOL) character such as a carriage return or line feed (CR or LF) and each field is separated by a designated character such as a comma.</p>

Field Name	Description
Character encoding	The text file's encoding. Select one of these: <ul style="list-style-type: none"> CP1252 This encoding is also known as the Windows-1252 or simply Windows character set. It is a super set of ISO-8859-1 and uses the 128-159 code range to display additional characters not included in the ISO-8859-1 character set. UTF-8 Supports all Unicode characters and is backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html. UTF-16 Supports all Unicode characters but is not backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html. US-ASCII A character encoding based on the order of the English alphabet. UTF-16BE UTF-16 encoding with big endian byte serialization (most significant byte first). UTF-16LE UTF-16 encoding with little endian byte serialization (least significant byte first). ISO-8859-1 An ASCII character encoding typically used for Western European languages. Also known as Latin-1. ISO-8859-3 An ASCII character encoding typically used for Southern European languages. Also known as Latin-3. ISO-8859-9 An ASCII character encoding typically used for Turkish language. Also known as Latin-5. CP850 An ASCII code page used to write Western European languages. CP500 An EBCDIC code page used to write Western European languages. Shift_JIS A character encoding for the Japanese language. MS932 A Microsoft's extension of Shift_JIS to include NEC special characters, NEC selection of IBM extensions, and IBM extensions. CP1047 An EBCDIC code page with the full Latin-1 character set.

Field Name	Description						
Field separator	<p>Specifies the character used to separate fields in a delimited file. For example, this record uses a pipe () as a field separator:</p> <pre>7200 13TH ST MIAMI FL 33144</pre> <p>These characters available to define as field separators are:</p> <ul style="list-style-type: none"> • Space • Tab • Comma • Period • Semicolon • Pipe <p>If the file uses a different character as a field separator, click the ellipses button to select another character as a delimiter.</p>						
Text qualifier	<p>The character used to surround text values in a delimited file. For example, this record uses double quotes (") as a text qualifier.</p> <pre>"7200 13TH ST" "MIAMI" "FL" "33144"</pre> <p>The characters available to define as text qualifiers are:</p> <ul style="list-style-type: none"> • Single quote (') • Double quote (") <p>If the file uses a different text qualifier, click the ellipses button to select another character as a text qualifier.</p>						
Record separator	<p>Specifies the character used to separate records in line a sequential or delimited file. This field is not available if you check the Use default EOL check box.</p> <p>The record separator settings available are:</p> <table border="0"> <tr> <td>Linux (U+000A)</td> <td>A line feed character separates the records. This is the standard record separator for Linux systems.</td> </tr> <tr> <td>Macintosh (U+000D)</td> <td>A carriage return character separates the records. This is the standard record separator for Macintosh systems.</td> </tr> <tr> <td>Windows (U+000D U+000A)</td> <td>A carriage return followed by a line feed separates the records. This is the standard record separator for Windows systems.</td> </tr> </table> <p>If your file uses a different record separator, click the ellipses button to select another character as a record separator.</p>	Linux (U+000A)	A line feed character separates the records. This is the standard record separator for Linux systems.	Macintosh (U+000D)	A carriage return character separates the records. This is the standard record separator for Macintosh systems.	Windows (U+000D U+000A)	A carriage return followed by a line feed separates the records. This is the standard record separator for Windows systems.
Linux (U+000A)	A line feed character separates the records. This is the standard record separator for Linux systems.						
Macintosh (U+000D)	A carriage return character separates the records. This is the standard record separator for Macintosh systems.						
Windows (U+000D U+000A)	A carriage return followed by a line feed separates the records. This is the standard record separator for Windows systems.						

Field Name	Description
Use default EOL	<p>Specifies that the file's record separator is the default end of line (EOL) character used on the operating system on which the Spectrum Technology Platform server is running.</p> <p>Do not select this option if the file uses an EOL character that is different from the default EOL character used on the server's operating system. For example, if the file uses a Windows EOL but the server is running on Linux, do not check this option. Instead, select the Windows option in the Record separator field.</p>
Record length	<p>For fixed width files, specifies the exact number of characters in each record.</p> <p>For line sequential files, specifies the length, in characters, of the longest record in the file.</p>
First row is header record	<p>Specifies whether the first record in a delimited file contains header information and not data.</p> <p>For example, this file snippet shows a header row in the first record.</p> <pre>"AddressLine1" "City" "StateProvince" "PostalCode" "7200 13TH ST" "MIAMI" "FL" "33144" "One Global View" "Troy" "NY" 12180</pre>
Treat records with fewer fields than defined as malformed	<p>Delimited file records containing fewer fields than are defined on the Fields tab will be treated as malformed.</p>
Import	<p>Imports the file layout definition, encoding setting, and sort options from a settings file. The settings file is created by exporting settings from another Read from File or Write to File stage that used the same input file or a file that has the same layout as the file you are working with.</p>
Export	<p>Saves the file layout definition, encoding setting, and sort options to a settings file. You can then import these settings into other Read from File or Write to File stages that use the same input file or a file that has the same traits as the file you are working with now. You can also use the settings file with job executor to specify file settings at runtime.</p> <p>For information about the settings file, see The File Definition Settings File on page 318.</p>

Fields Tab

The Fields tab defines the names, positions, and, for fixed width and line sequential files, lengths of fields in the file. For more information, see these topics:

[Defining Fields In a Delimited Output File](#) on page 404

[Defining Fields In a Line Sequential or Fixed Width File](#) on page 406

Sort Fields Tab

The Sort Fields tab defines fields by which to sort the output records before they are written to the output file. Sorting is optional. For more information, see [Sorting Output Records](#) on page 409.

Runtime Tab

Option Name	Description
File name	This displays the file defined on the File Properties tab.
Generate multiple files	<p>Select this option to write records to different files instead of writing all records to one file. The file to which each record is written is specified in the record itself. Each record must contain a field that specifies either a file name or the full file path of the file to which you want the record written. For example, if you want to send the stock prices of different companies (of various groups) to all the clients separately, this feature writes the stock prices of different companies into separate files that may be sent to each of the clients, if you so wish. If you enable the Generate multiple file option you must specify an output file on either the Spectrum Technology Platform server or on an FTP server. If you want to write data to a file on an FTP server you must define a connection to the file server using Spectrum Management Console.</p> <p>Note: The records in the column you select in the File path field must be in sorted order. Use this feature when record contains either a file name or the full file path of the file.</p>
File path field	Selects the field that contains the path (either a file name or the full file path) of the file to which you want to write the record. This field is only enabled if you select Generate multiple files .

Option Name	Description
Write Mode	Specifies whether to add the dataflow's output to the end of the file or to delete the existing data in the file before writing the output:
Overwrite	Replaces the existing data in the output file each time the dataflow runs.
Append	Adds the dataflow's output to the end of the file without erasing the file's existing data.

Defining Fields In a Delimited Output File

In the Write to File stage, the **Fields** tab defines the names, position, and, for some file types, lengths of the fields in the file. After you define an output file on the **File Properties** tab you can define the fields.

If the output file contains a header record, you can quickly define the fields by clicking **Regenerate**.

To define fields with default values for position, length, and data type, click **Quick Add** and select the fields to add.

If the input file does not contain a header record, or if you want to manually define the fields, follow these steps:

1. Click **Add**.
2. In the **Name** field, choose the field you want to add.
3. In the **Type** field, select the data type of the field coming from the dataflow.

Spectrum Technology Platform supports these data types:

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.

bytearray An array (list) of bytes.

Note: Bytearray is not supported as an input for a REST service.

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 $(2-2^{-52}) \times 2^{1023}$. 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 $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -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).
list	Strictly speaking, a list is not a data type. However, when a field contains hierarchical data, it is treated as a "list" field. In Spectrum Technology Platform a list is a collection of data consisting of multiple values. For example, a field Names may contain a list of name values. This may be represented in an XML structure as: <pre data-bbox="418 779 1424 934" style="background-color: #f0f0f0; padding: 10px;"> <Names> <Name>John Smith</Name> <Name>Ann Fowler</Name> </Names></pre>
	It is important to note that the Spectrum Technology Platform list data type is different from the XML schema list data type in that the XML list data type is a simple data type consisting of multiple values, whereas the Spectrum Technology Platform list data type is similar to an XML complex data type.
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.

4. If you selected a date, time, or numeric data type, you can use the default date/time or number format or you can specify a different format for this specific field. The default format is either the system default format that has been set in the type conversion options in Spectrum Management Console, or it is the dataflow's default format specified in the type conversion options in Spectrum Enterprise Designer. The format that is in effect is displayed. To use the default format, leave **Default** selected. To specify a different format, choose **Custom** and follow these steps:
 - a) In the **Locale** field, select the country whose formatting convention you want to use. Your selection will determine the default values in the **Format** field. For date data, 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 **Format** field, select the format for the data. The format depends on the data type of the field. A list of the most commonly used formats for the selected locale is provided.

An example of the selected format is displayed to the right of the **Format** field.

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 32. To specify your own number format, type the format into the file using the notation described in [Number Patterns](#) on page 34.

5. Click **Add**.

After defining the fields in your output file, you can edit its contents and layout.

Option Name	Description
Add	Adds a field to the output. You can append a field to the end of the existing layout, or you can insert a field into an existing position and the position of the remaining fields will be adjusted accordingly.
Modify	Modifies the field's name and type.
Remove	Removes the selected field from the output.
Move Up/Move Down	Reorders the selected field.

Defining Fields In a Line Sequential or Fixed Width File

In the Write to File stage, the **Fields** tab defines the names, position, and, for some file types, lengths, of the fields in the file. After you define an output file on the **File Properties** tab you can define the fields.

To define fields with default values for position, length, and data type, click **Quick Add** and select the fields to add.

To add fields manually from a list of fields used in the dataflow, follow this procedure:

1. Click **Add**.
2. In the **Name** field, choose the field you want to add.
3. In the **Type** field, select the data type of the field coming from the dataflow.

Spectrum Technology Platform supports these data types:

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.

bytearray An array (list) of bytes.

Note: Bytearray is not supported as an input for a REST service.

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 $(2-2^{-52}) \times 2^{1023}$. 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 $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -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).

list Strictly speaking, a list is not a data type. However, when a field contains hierarchical data, it is treated as a "list" field. In Spectrum Technology Platform a list is a collection of data consisting of multiple values. For example, a field Names may contain a list of name values. This may be represented in an XML structure as:

```
<Names>
  <Name>John Smith</Name>
  <Name>Ann Fowler</Name>
</Names>
```

It is important to note that the Spectrum Technology Platform list data type is different from the XML schema list data type in that the XML list data type is a simple data type consisting of multiple values, whereas the Spectrum Technology Platform list data type is similar to an XML complex data type.

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.

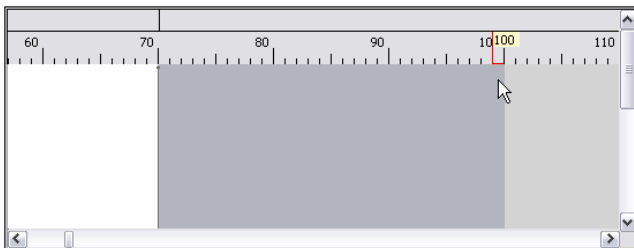
4. If you selected a date, time, or numeric data type, you can use the default date/time or number format or you can specify a different format for this specific field. The default format is either the system default format that has been set in the type conversion options in Spectrum Management Console, or it is the dataflow's default format specified in the type conversion options in Spectrum Enterprise Designer. The format that is in effect is displayed. To use the default format, leave **Default** selected. To specify a different format, choose **Custom** and follow these steps:
 - a) In the **Locale** field, select the country whose formatting convention you want to use. Your selection will determine the default values in the **Format** field. For date data, 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 **Format** field, select the format for the data. The format depends on the data type of the field. A list of the most commonly used formats for the selected locale is provided.

An example of the selected format is displayed to the right of the **Format** field.

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 32. To specify your own number format, type the format into the file using the notation described in [Number Patterns](#) on page 34.

5. The **Start Position** and **Length** fields are automatically filled in based on the data in the dataflow and number of fields you have already added.
6. Click **Add**.

Alternatively, you can also add a field by first defining the starting position and length of the field. To do this, under **Sample File** click at the position where you want to begin a field and drag to the left so that the desired field is highlighted, as shown here:



After defining the fields in your output file, you can edit its contents and layout. The **Recalculate start position** option tells the Write to File stage to recalculate the positions of the fields when you modify, move, or remove a field in the output file. Uncheck this box if you do not want the positions recalculated and instead want the fields to stay in their existing position after you edit the output file.

Option Name	Description
Add	Adds a field to the output.
Modify	Modifies the field's name, type, start position, and length.
Remove	Removes the selected field from the output.
Move Up/Move Down	Reorders the selected field.

Sorting Output Records

In the Write to File stage, the **Sort Fields** tab defines fields by which to sort the output records before they are written to the output file. Sorting is optional.

1. In Write to File, click the **Sort Fields** tab.
2. Click **Add**.
3. Click the drop-down arrow in the **Field Name** column and select the field you want to sort by. The fields available for selection depend on the fields in the dataflow.
4. In the **Order** column, select Ascending or Descending.
5. Repeat until you have added all the output fields you wish to use for sorting. Change the order of the sort by highlighting the row for the field you wish to move and clicking **Up** or **Down**.
6. Default sort performance options for your system are set in Spectrum Management Console. If you want to override your system's default sort performance options, click **Advanced**. The **Advanced Options** dialog box contains these sort performance options:

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:

$$\frac{(NumberOfRecords \times 2)}{InMemoryRecordLimit} = NumberOfTempFilesN$$

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: $(InMemoryRecordLimit \times MaxNumberOfTempFiles \div 2) \geq TotalNumberOfRecords$

The File Definition Settings File

A file definition settings file contains the file layout, encoding, and sort options that have been exported from a Read from File or Write to File stage. The file definitions settings file can be imported into Read from File or Write to File to quickly set the stage's options instead of manually specifying the options.

The easiest way to create a file definition settings file is to use specify the file settings using Read from File or Write to File, then click the **Export** button to generate the file definitions settings file.

However, for your information the schema of the file definition settings file is shown below. Each element in the XML file has a type, and if that type is anything other than string or integer, the acceptable values are shown. These values correspond directly to options in the stage's dialog box. For example, the FileTypeEnum element corresponds to the Record Type field on the File Properties tab, and these values appear in the schema: linesequential, fixedwidth, and delimited.

Note: If you enter "custom" for the LineSeparator, FieldSeparator or TextQualifier fields, a corresponding custom element must also be included (for example, "CustomLineSeparator", "CustomFieldSeparator", or "CustomTextQualifier") with a hexadecimal number representing the character, or sequence of characters, to use.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="FileSchema" nillable="true" type="FileSchema"/>
  <xs:complexType name="FileSchema">
```

```

<xs:sequence>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    default="linesequential"
    name="Type"
    type="FileTypeEnum"/>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    default="UTF-8" name="Encoding" type="xs:string"/>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    name="RecordLength"
    type="xs:int"/>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    default="default"
    name="LineSeparator"
    type="LineSeparatorEnum"/>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    name="CustomLineSeparator"
    type="xs:string"/>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    default="comma"
    name="FieldSeparator"
    type="FieldSeparatorEnum"/>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    name="CustomFieldSeparator"
    type="xs:string"/>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    default="none"
    name="TextQualifier"
    type="TextQualifierEnum"/>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    name="CustomTextQualifier"
    type="xs:string"/>
  <xs:element
    minOccurs="0"
    maxOccurs="1"
    default="false"

```

```

        name="HasHeader"
        type="xs:boolean"/>
<xs:element
  minOccurs="0"
  maxOccurs="1"
  default="true"
  name="EnforceColumnCount"
  type="xs:boolean"/>
<xs:element
  minOccurs="0"
  maxOccurs="1"
  name="Fields"
  type="ArrayOfFieldSchema"/>
</xs:sequence>
</xs:complexType>
<xs:simpleType name="FileTypeEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="linesquential"/>
    <xs:enumeration value="fixedwidth"/>
    <xs:enumeration value="delimited"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="LineSeparatorEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="default"/>
    <xs:enumeration value="windows"/>
    <xs:enumeration value="linux"/>
    <xs:enumeration value="mac"/>
    <xs:enumeration value="custom"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="FieldSeparatorEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="comma"/>
    <xs:enumeration value="tab"/>
    <xs:enumeration value="space"/>
    <xs:enumeration value="semicolon"/>
    <xs:enumeration value="period"/>
    <xs:enumeration value="pipe"/>
    <xs:enumeration value="custom"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="TextQualifierEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="none"/>
    <xs:enumeration value="single"/>
    <xs:enumeration value="double"/>
    <xs:enumeration value="custom"/>
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="ArrayOfFieldSchema">
  <xs:sequence>
    <xs:element

```

```

        minOccurs="0"
        maxOccurs="unbounded"
        name="Field"
        nillable="true"
        type="FieldSchema"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="FieldSchema">
    <xs:sequence>
        <xs:element
            minOccurs="0"
            maxOccurs="1"
            name="Name"
            type="xs:string"/>
        <xs:element
            minOccurs="0"
            maxOccurs="1"
            default="string"
            name="Type"
            type="xs:string"/>
        <xs:element
            minOccurs="1"
            maxOccurs="1"
            name="Position"
            type="xs:int"/>
        <xs:element
            minOccurs="0"
            maxOccurs="1"
            name="Length"
            type="xs:int"/>
        <xs:element
            minOccurs="0"
            maxOccurs="1"
            default="false"
            name="Trim"
            type="xs:boolean"/>
        <xs:element
            minOccurs="0"
            maxOccurs="1"
            name="Locale"
            type="Locale"/>
        <xs:element
            minOccurs="0"
            maxOccurs="1"
            name="Pattern"
            type="xs:string"/>
        <xs:element
            minOccurs="0"
            maxOccurs="1"
            default="none"
            name="Order"
            type="SortOrderEnum"/>
    </xs:sequence>

```

```

</xs:complexType>
<xs:complexType name="Locale">
  <xs:sequence>
    <xs:element
      minOccurs="0"
      maxOccurs="1"
      name="Country"
      type="xs:string"/>
    <xs:element
      minOccurs="0"
      maxOccurs="1"
      name="Language"
      type="xs:string"/>
    <xs:element
      minOccurs="0"
      maxOccurs="1"
      name="Variant"
      type="xs:string"/>
  </xs:sequence>
</xs:complexType>
<xs:simpleType name="SortOrderEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="none"/>
    <xs:enumeration value="ascending"/>
    <xs:enumeration value="descending"/>
  </xs:restriction>
</xs:simpleType>
</xs:schema>

```

Configuring Dataflow Options

This procedure describes how to configure a dataflow to support runtime options for Write to File stage.

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. Expand the **Write to File** stage.

The Dataflow options exposed are:

- a. Character Encoding
- b. Field Separator
- c. Text Qualifier
- d. Record Length
- e. First Row is Header Record

6. The selected Write to File option name is displayed in **Option name** and **Option label** fields. This is the option name that will have to be specified at run time in order to set this option.
7. Enter a description of the option in the **Description** field.
8. In the **Target** field, select the option **Selected stage(s)**.
9. 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.
10. 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 845.

11. Click **OK**.
12. Continue adding options as desired.
13. Click **OK** in the **Dataflow Options** dialog box when you are done adding options.
14. Save and expose the dataflow.

Dataflow Options Rules

1. *Character Encoding:* All encoding types that are valid for the underlying JVM are accepted. This option cannot be empty.
2. *Field Separator:* Any single character delimiter is accepted. Currently, HEX values and spaces are not supported.
3. *Text Qualifier:* Any single character qualifier is accepted. HEX values are not supported.
4. *Record Length:* Only integers accepted. Cannot be blank or non numeric.
5. *Starting Record:* Only integers accepted. Cannot be non numeric.
6. *Max records:* Only integers accepted. Cannot be non numeric.
7. *First Row is Header:* Only boolean values of `true` and `false` accepted. Cannot be blank.

Write to Hadoop Sequence File

The **Write to Hadoop Sequence File** stage writes data to a sequence file as output from a dataflow. A sequence file is a flat file consisting of binary key/value pairs. For more information, go to wiki.apache.org/hadoop/SequenceFile.

The stage supports reading data from and writing data to HDFS 3.x. The support includes:

- Connectivity to HDFS from Spectrum on Windows
- Support and connectivity to Hadoop 3.x from Spectrum with high availability
- Kerberos-enabled HDFS connectivity through Windows

Also see [Configuring HDFS Connection for HA Cluster](#) and [Best Practices for connecting to HDFS 3.x and Hive 2.1.1](#).

Note: The Write to Hadoop Sequence File stage only supports delimited, uncompressed sequence files located on Hadoop Distributed File System (HDFS).

Related tasks: [Connecting to Hadoop](#): To be able to use **Write to Hadoop Sequence File** stage, you need to create a connection to the Hadoop file server. Once you do that, the name by which you save the connection is displayed as the server name.

File Properties Tab

Fields	Description
Field separator	<p>Specifies the character used to separate fields in a delimited file. For example, this record uses a pipe () as a field separator:</p> <pre>7200 13TH ST MIAMI FL 33144</pre> <p>These characters available to define as field separators are:</p> <ul style="list-style-type: none"> • Space • Tab • Comma • Period • Semicolon • Pipe <p>If the file uses a different character as a field separator, click the ellipses button to select another character as a delimiter.</p>
Text qualifier	<p>The character used to surround text values in a delimited file. For example, this record uses double quotes (") as a text qualifier.</p> <pre>"7200 13TH ST" "MIAMI" "FL" "33144"</pre> <p>The characters available to define as text qualifiers are:</p> <ul style="list-style-type: none"> • Single quote (') • Double quote (") <p>If the file uses a different text qualifier, click the ellipses button to select another character as a text qualifier.</p>

Fields Tab

The Fields tab defines the names, positions, and types of fields in the file. For more information, see [Defining Fields In an Output Sequence File](#) on page 417

Defining Fields In an Output Sequence File

In the Write to Hadoop Sequence File stage, the **Fields** tab defines the names, positions, and types of fields in the file. After you define an input file on the **File Properties** tab you can define the fields.

1. To select the desired fields from the input data, or an existing file, click **Quick Add**.
 - a) Select the specific fields from the input data.
 - b) Click **OK**.
2. To add new fields, click **Add**.
 - a. Enter the Name of the field.
 - b. Select the Type of the field.

The stage supports the following data types:

double	A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. 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 $(2-2^{-23}) \times 2^{127}$. 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.

3. If you're overwriting an existing file, click **Regenerate** to pick the schema from the existing file, and modify it.
This generates the schema based on the first 50 records in the input data to this stage.
4. If you want to have any excess space characters removed from the beginning and end of a field's character string, select the **Trim Spaces** check box.
5. Specify an option to generate the key:

Auto Generate The Hadoop cluster auto generates the key. For auto generation, all the fields in the grid are considered value fields. The data type of the key is long.

User Defined By default, the first field in the grid is selected as the key field. An icon is displayed to indicate that the field is the key field. You can select any other field as the key field.

After defining the fields in your output file, you can edit its contents and layout.

Option Name	Description
Add	Adds a field to the output. You can append a field to the end of the existing layout, or you can insert a field into an existing position and the position of the remaining fields will be adjusted accordingly.
Modify	Modifies the field's name and type.
Remove	Removes the selected field from the output.
Move Up/Move Down	Reorders the selected field.

Write to Hive File

The **Write to Hive File** stage writes the dataflow input to the specified output Hive file.

You can select any of these supported Hive file formats for the output file: ORC, Parquet, and Avro.

The stage supports reading data from and writing data to HDFS 3.x. The support includes:

- Connectivity to HDFS and Hive from Spectrum on Windows
- Support and connectivity to Hadoop 3.x from Spectrum with high availability
- Kerberos-enabled HDFS connectivity through Windows
- Support of Datetime datatype in the Parquet file format

Also see [Configuring HDFS Connection for HA Cluster](#) and [Best Practices for connecting to HDFS 3.x and Hive 2.1.1](#).

Related task:

Connecting to Hadoop: To be able to use **Write to Hive File** stage, you need to create a connection to the Hadoop file server. Once you do that, the name by which you save the connection is displayed as the server name.

*File Properties tab***Table 32: Common File Properties**

Fields	Description
Server name	Indicates that the file selected in the File name field is located on the Hadoop system. Once you select a file located on a Hadoop system, the Server name reflects the name of the respective file server, as specified in Spectrum Management Console.
File name	Click the ellipses button (...) to browse to the output Hive file to be created in the defined Hadoop file server. The output data of this stage is written to the selected file. Note: You need to create a connection to the Hadoop file server in Spectrum Management Console before using it in the stage.
File type	Select one of these supported Hive file formats: <ul style="list-style-type: none"> • ORC • Parquet • Avro

Table 33: ORC File Properties

Fields	Description
Buffer size	Defines the buffer size to be allocated while writing to an ORC file. This is specified in kilobytes. Note: The default buffer size is 256 KB.
Stripe size	Defines the size of stripes to be created while writing to an ORC file. This is specified in megabytes. Note: The default stripe size is 64 MB.
Row index stride	Defines the number of rows to be written between two consecutive row index entries. Note: The default Row Index Stride is 10000 rows.

Fields	Description
Compression type	<p>Defines the compression type to be used while writing to an ORC file. The compression types available are <i>ZLIB</i> and <i>SNAPPY</i>.</p> <p>Note: The default compression type is <i>ZLIB</i>.</p>
Padding	<p>Indicates whether the stripes are padded to minimize stripes that cross HDFS block boundaries, while writing to an ORC file.</p> <p>Note: By default, the Padding checkbox is selected.</p>
Preview	<p>The first 50 records of the written file are fetched and displayed in the Preview grid, after the dataflow is run at least once and the data has been written to the selected file.</p>

Table 34: Parquet File Properties

Fields	Description
Compression type	<p>Defines the compression type to be used while writing to a PARQUET file. The compression types available are <i>UNCOMPRESSED</i>, <i>GZIP</i> and <i>SNAPPY</i>.</p> <p>Note: The default compression type is <i>UNCOMPRESSED</i>.</p>
Block size	<p>Defines the size of block to be created while writing to a PARQUET file. This is specified in megabytes.</p> <p>Note: The default block size is 128 MB.</p>
Page size	<p>The page size is for compression. When reading, each page can be decompressed independently. This is specified in kilobytes.</p> <p>Note: The default page size is 1024 KB.</p>
Enable dictionary	<p>To enable/disable dictionary encoding.</p> <p>Attention: The dictionary must be enabled for the Dictionary Page Size to be enabled.</p> <p>Note: The default is <i>true</i>.</p>

Fields	Description
Dictionary Page size	<p>There is one dictionary page per column per row group when dictionary encoding is used. The dictionary page size functions like the page size. This is specified in kilobytes.</p> <p>Note: The default dictionary Page size is 1024 KB.</p>
Writer version	<p>Parquet supports two writer API versions: <code>PARQUET_1_0</code> and <code>PARQUET_2_0</code>.</p> <p>Note: The default is <code>PARQUET_1_0</code>.</p>
Preview	<p>The first 50 records of the written file are fetched and displayed in the Preview grid, after the dataflow is run at least once and the data has been written to the selected file.</p>

Table 35: Avro File Properties

Fields	Description
Sync Interval (in Bytes)	<p>Specifies the approximate number of uncompressed bytes to be written in each block. The valid values range from 32 to 2^{30}. However, it is suggested to keep the sync interval between 2K and 2M.</p> <p>Note: The default sync interval is 16000.</p>
Compression	<p>Defines the compression type to be used while writing to an Avro file. The compression types available are NONE, SNAPPY and DEFLATE. Choosing DEFLATE compression gives you an additional option of selecting the compression level (described below).</p> <p>Note: The default compression type is <code>NONE</code>.</p>
Compression level	<p>This field is displayed if you select the <code>DEFLATE</code> option in the Compression field above.</p> <p>It can have values ranging from 0 to 9, where 0 denotes no compression. The compression level increases from 1 to 9, with a simultaneous increase in the time taken to compress the data.</p> <p>Note: The default compression level is 1.</p>
Preview	<p>The first 50 records of the written file are fetched and displayed in this grid, after the dataflow is run at least once and the data is written to the selected file.</p>

Fields tab

The **Fields** tab defines the names and types of the fields as present in the source file of this stage, and to be selected to be written to the output file.

For more information, see [Defining Fields for Writing to Hive File](#) on page 422.

Runtime tab

The **Runtime** tab provides the option to **Overwrite** an existing file of the same name in the configured Hadoop file server. If you check the **Overwrite** checkbox, then on running the dataflow, the new output Hive file overwrites any existing file of the same name in the same Hadoop file server.

By default, the **Overwrite** checkbox is unchecked.

Note: If you do not select **Overwrite**, an exception is thrown while running the dataflow, if the file to be written has the same name as an existing file in the same Hadoop file server.

Defining Fields for Writing to Hive File

In the **Fields** tab of the **Write to Hive File** stage, the schema names and datatypes of the fields in the input data to the stage are listed.

1. To select the desired fields from the input data, or an existing file, click **Quick Add**.
 - a) Select the specific fields from the input data.
 - b) Click **OK**.
2. To add new fields, click **Add**.
 - a) Enter the **Name** of the field.
 - b) Select the **Type** of the field. The stage supports these data types:

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.

Note: In Parquet files, `datetime` and `time` datatypes are mapped as `String`.

double A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52}) \times 2^{1023}$. 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 $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -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).

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.

Note: For Avro and Parquet Hive files, the `bigdecimal` datatype is converted to a `decimal` datatype with precision 38 and scale 10.

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.

- c) In the **Position** field, enter the position of this field within the record.

For example, in this input file, AddressLine1 is in position 1, City is in position 2, StateProvince is in position 3, and PostalCode is in position 4.

```
"AddressLine1"|"City"|"StateProvince"|"PostalCode"
"7200 13TH ST"|"MIAMI"|"FL"|"33144"
"One Global View"|"Troy"|"NY"|"12180"
```

3. If you're overwriting an existing file, click **Regenerate** to pick the schema from the existing file, then modify it.

This generates the schema based on the metadata of the existing file, in case of ORC and Parquet output files.

The **Name** column lists the names of the various columns of the input data. The **Type** column lists the datatypes of each respective field of the input data.

Note: In case of *Parquet* file type, another column **Nullable** indicates whether the field is nullable or not. You can check this checkbox for a particular field to make the field nullable, or uncheck it otherwise.

4. You can modify the names, datatypes and sequence of the selected columns in the output using these buttons:

Option Name	Description
Add	Adds a field to the output.

Option Name	Description
Modify	Modifies the selected field's name and datatype.
Remove	Removes the selected field from the output.
Move Up/Move Down	Reorders the position of the selected field in the output.

- Click **OK**.

Write to NoSQL DB

The **Write to NoSQL DB** writes the output of a dataflow to a NoSQL database. The stage supports MongoDB and Couchbase database (version 5.x and above).

Note: Significant performance improvements can be achieved by using multiple runtime instances of Write to NoSQLDB. To specify multiple runtime instances, click the **Runtime** button.

General Tab

Fields	Description
Connection	<p>Select the required database connection from the dropdown list. The options displayed are based on the connections defined in Spectrum Management Console.</p> <p>To add a new connection, see Connecting to NoSQL.</p> <p>To modify an existing connection, select and open it from the list of connections on the Connections page of Spectrum Management Console, make the required updates, and click the Save button.</p>
Table/View	<p>Specify the name of the collection you want to write to.</p> <p>You can create a new collection in the NoSQL database by entering a collection name in the Table/View drop box and clicking Create.</p> <p>Note: For Couchbase the 'table/view drop down' and 'create' button is disabled, as we write to a bucket instead of a view. In addition, the Preview button is also disabled.</p>

Fields	Description
Ignore NULL Values	<p>If this option is enabled, any field that has a NULL value, is ignored.</p> <p>Note: If you do not enable this option, any field that has a NULL value will be also written to the database.</p>
Preview	<p>Displays the records from the selected table.</p> <p>Note: For MongoDB data sources, clicking Preview shows the filtered records, if one or more <code>where</code> clauses have been entered in the Where field. If no where clause has been entered, the preview displays all the records.</p> <p>Note: For Couchbase data sources, clicking Preview also displays the added <code>_id</code> field containing the key. If the record already has an <code>_id</code> field, then the added <code>_id</code> field overwrites the pre-existing one at the time of previewing the fields.</p>
Expand All	Expands the items in the preview tree.
Collapse All	Collapses the items in the preview tree.

Fields Tab

The Fields tab allows you to select the data that you want to write to the database. For more information, see [Defining Fields in a NoSQL Database](#) on page 425.

Defining Fields in a NoSQL Database

In the Write to NoSQL DB stage, the **Fields** tab defines the names and types of fields fetched from the previous stage.

1. On the **Fields Tab**, click **Regenerate**.

This displays the fields that are flowing from the previous stage.

The data is displayed in the following format: `Fieldname (datatype)`.

Note: For Couchbase, if the record has an `_id` field, this field will be used as a key for writing the record into the database. Else, the key for the record, will be autogenerated and written to the database.

2. To modify the name and type of a field, highlight the field, and click **Modify**.
3. In the **Name** field, choose the field you want to add or type the name of the field.
4. In the **Type** field, you can leave the data type as string if you do not intend to perform any mathematical operations with the data. However, if you intend to perform these kinds of operations,

select an appropriate data type. This will convert the string data from the file to a data type that will enable the proper manipulation of the data in the dataflow.

The stage supports the following data types:

double	A numeric data type that contains both negative and positive double precision numbers between 2^{-1074} and $(2-2^{-52})\times 2^{1023}$. 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 $(2-2^{-23})\times 2^{127}$. In E notation, the range of values is -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.

- From the **Fields** tab, you can either individually select each field to be written to the database or click **Select All** to select all the fields.
- Optionally, from the **Runtime** tab, you can set the batch size. The batch size denotes the number of records that are to be written to the database at one time.
- Click **OK**.

NoSQL DB Dataflow Options

This procedure describes how to configure a dataflow to support runtime options for NoSQL DB.

- Open the flow in Spectrum Enterprise Designer.
- If you want to configure runtime options for a stage in an embedded flow, open the embedded flow.
- Click the Dataflow Options icon on the toolbar or click **Edit > Dataflow Options**. The **Dataflow Options** dialog box appears.
- Click **Add**. The **Define Dataflow Options** dialog box appears.
- Expand the NoSQLDB stage.
- The Dataflow options are exposed as described in the following table:

Database	Read	Write
Mongo DB	Connection	Connection
	Table	Table
Couchbase DB	Connection	Connection
	View	

Database	Read	Write
	Design Document Name	

The selected NoSQL DB option name is displayed in **Option name** and **Option label** fields. This is the option name that will have to be specified at run time in order to set this option.

7. Enter a description of the option in the **Description** field.
8. In the **Target** field, select the option **Selected stage(s)**.
9. 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.
10. 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 845.

11. Click **OK**.
12. Continue adding options as desired.
13. Click **OK** in the **Dataflow Options** dialog box when you are done adding options.
14. Save and expose the dataflow.

Write to Spreadsheet

Write to Spreadsheet writes data to an Excel spreadsheet as output from a dataflow.

File Properties Tab

The **File Properties** tab contains options for specifying the spreadsheet and data to write from a dataflow.

Field Name	Description
Server name	Indicates the file you select in the File name field is located on the Spectrum Technology Platformserver.
File name	Specifies the path to the file. Click the ellipses button (...) to browse to the file. Attention: If the Spectrum Technology Platform server is running on Linux, remember that file names and paths on these platforms are case sensitive.

Field Name	Description
Write Mode	<p>Specifies how you want to write data in the spreadsheet in order to write into the dataflow. You can create a spreadsheet at runtime using any of the following options. Options are as following:</p> <p>Create or Overwrite Creates a new file and replaces the existing data in the output file each time the dataflow runs.</p> <p>Insert Adds the dataflow's output to the mapped area and shifts the data down if already present there.</p> <p>Append Adds the dataflow's output to the end of the file without erasing the file's existing data.</p>
Sheet name	Specifies a sheet name in the spreadsheet to which you want to write data into the dataflow.
Start writing from	Specifies either a row-column combination (A1 or B2 ..) or a column from where you want the data to be written. For the option Insert you need to provide both row & column. For the option Append you can only provide column as it ignores the row value.
First row as column header	Specifies the first row in a file contains header information and not data.

Fields Tab

The Fields tab defines columns, positions, types and nullable values for the fields in the file. For more information, see the following topic:

[Defining fields in an Output file](#) on page 428

Defining fields in an Output file

In Write to Spreadsheet, the **Fields** tab defines the names, position, and data types of the fields in the file. After you define an output file on the **File Properties** tab you can define the fields. If the option **Nullable** is checked and the **Name** field contains a null value, then the dataflow will write the null value in the spreadsheet.

If the output file contains a header record, you can quickly define the fields by clicking **Regenerate**.

To define fields with default values for position, length, and data type, click **Quick Add** and select the fields to add.

If the input file does not contain a header record, or if you want to manually define the fields, follow these steps:

1. Click **Add**.

2. In the **Name** field, choose the field you want to add.
3. In the **Type** field, select the data type of the field coming from the dataflow.

Spectrum Technology Platform supports the following data types:

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.
bytearray	An array (list) of bytes. Note: Bytearray is not supported as an input for a REST service.
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 $(2-2^{-52}) \times 2^{1023}$. 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 $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -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.

4. Click **Add**.

After defining the fields in your output file, you can edit its contents and layout.

Option Name	Description
Add	Adds a field to the output. You can append a field to the end of the existing layout, or you can insert a field into an existing position and Write to Spreadsheet will adjust the remaining fields accordingly.
Modify	Modifies the field's name and type.
Remove	Removes the selected field from the output.
Move Up/Move Down	Reorders the selected field.

Write to Variable Format File

Write to Variable Format File writes records of varying layout to a file.

Variable format files have these characteristics:

- Records in the file may have different fields, and different numbers of fields.
- Each record must contain a tag (usually a number) identifying the type of record.
- Hierarchical relationships are supported.

Example of a Variable Format File

This example shows a variable format file containing information about checking account activity for two customers, Joe Smith and Anne Johnson. In this example, the file is a delimited file that uses a comma as the field delimiter.

```
001 Joe,Smith,M,100 Main St,555-234-1290
100 CHK12904567,12/2/2007,6/1/2012,CHK
200 1000567,1/5/2012,Fashion Shoes,323.12
001 Anne,Johnson,F,1202 Lake St,555-222-4932
100 CHK238193875,1/21/2001,4/12/2012,CHK
200 1000232,3/5/2012,Blue Goose Grocery,132.11
200 1000232,3/8/2012,Trailway Bikes,540.00
```

The first field in each record contains the tag which identifies the type of record and therefore the record's format:

- 001: Customer record
- 100: Account record
- 200: Account transaction record

For delimited files it is common for the tag value (001, 100, 200) to be in a fixed number of bytes at the start of the record as shown in the above example.

Each record has its own format:

- 001: FirstName,LastName,Gender,Address,PhoneNumber
- 100: AccountID,DateOpened,ExpirationDate,TypeOfAccount
- 200: TransactionID,DateOfTransaction,Vendor,Amount

Record format 100 (account record) is a child of the previous 001 record, and record format 200 (account transaction record) is a child of the previous record 100 (account record). So in the example file, Joe Smith's account CHK12904567 had a transaction on 1/5/2012 in the amount of 323.12 at Fashion Shoes. Likewise, Anne Johnson's account CHK238193875 had two transactions, one on 3/5/2012 at Blue Goose Grocery and one on 3/8/2012 at Trailway Bikes.

File Properties Tab

Option Name	Description
Server name	Indicates whether the file you select as input is located on the computer running Spectrum Enterprise Designer or on the Spectrum Technology Platform server. If you select a file on the local computer, the server name will be My Computer. If you select a file on the server the server name will be Spectrum Technology Platform.
File name	Specifies the path to the file. Click the ellipses button (...) to locate the file you want. Note: If the Spectrum Technology Platform server is running on Linux, remember that file names and paths on these platforms are case sensitive.
Root tag name	The tag to use for records that are a parent of other record types. For example if you have three record types 001, 100, and 200, and record types 100 and 200 are children of record type 001, then 001 is the root tag.
Use fixed-width tags	Specifies whether to allocate a fixed amount of space at the beginning of each record in which to place the record tag. This example shows a file with the tags 001, 100, and 200 in a fixed-width field: <pre>001 Joe,Smith,M,100 Main St,555-234-1290 100 CHK12904567,12/2/2007,6/1/2012,CHK 200 1000567,1/5/2012,Mike's Shoes,323.12</pre>

Option Name	Description
Tag width	<p>If you check the Use fixed-width tags box, this option specifies the number of spaces to allocate for tags at the beginning of each record. For example, if you specify 7, then the first seven positions in each record will be reserved for the tag. The value you specify must be greater than or equal to the size in characters of the longest tag name. For information about tag names, see Tag Names in Variable Format Files on page 439.</p> <p>The value in the Tag width field is automatically increased if you add fields on the Fields tab that have a name that is longer than the value specified.</p> <p>The maximum tag width is 1024.</p>
Remove numeric tag prefix	<p>Removes the "NumericTag_" portion of the field name before writing the tag to the file. The "NumericTag_" prefix is added to tag names by the Read from Variable Format File stage for any tag names that start with a number. This is because the tag name is used as the name of a list dataflow field which contains the data in the record, and dataflow field names cannot begin with a number. For example, a tag 100 would be changed to list field named "NumericTag_100". If you enable this option, this field would be written to the output file as a record with a tag of "100" instead of "NumbericTag_100".</p>

Option Name	Description
Character encoding	The text file's encoding. Select one of these:
CP1252	This encoding is also known as the Windows-1252 or simply Windows character set. It is a super set of ISO-8859-1 and uses the 128-159 code range to display additional characters not included in the ISO-8859-1 character set.
UTF-8	Supports all Unicode characters and is backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html .
UTF-16	Supports all Unicode characters but is not backwards-compatible with ASCII. For more information about UTF, see unicode.org/faq/utf_bom.html .
US-ASCII	A character encoding based on the order of the English alphabet.
UTF-16BE	UTF-16 encoding with big endian byte serialization (most significant byte first).
UTF-16LE	UTF-16 encoding with little endian byte serialization (least significant byte first).
ISO-8859-1	An ASCII character encoding typically used for Western European languages. Also known as Latin-1.
ISO-8859-3	An ASCII character encoding typically used for Southern European languages. Also known as Latin-3.
ISO-8859-9	An ASCII character encoding typically used for Turkish language. Also known as Latin-5.
CP850	An ASCII code page used to write Western European languages.
CP500	An EBCDIC code page used to write Western European languages.
Shift_JIS	A character encoding for the Japanese language.
MS932	A Microsoft's extension of Shift_JIS to include NEC special characters, NEC selection of IBM extensions, and IBM extensions.
CP1047	An EBCDIC code page with the full Latin-1 character set.

Option Name	Description
Field separator	<p>Specifies the character used to separate fields in a delimited file. For example, this record uses a pipe () as a field separator:</p> <pre data-bbox="565 415 1040 447">7200 13TH ST MIAMI FL 33144</pre> <p>These characters available to define as field separators are:</p> <ul data-bbox="548 535 683 730" style="list-style-type: none"> • Space • Tab • Comma • Period • Semicolon • Pipe <p>If the file uses a different character as a field separator, click the ellipses button to select another character as a delimiter.</p>
Tag separator	<p>Specifies the character placed after the tag field to demarcate the identifying field for each record in a delimited file. A tag separator must be a single character.</p> <p>By default, these characters are available to be selected as tag separators:</p> <ul data-bbox="548 1003 683 1199" style="list-style-type: none"> • Space • Tab • Comma • Period • Semicolon • Pipe <p>If the file uses a different character as a tag separator, click the ellipses button to add and select a custom tag separator.</p> <p>Note: By default, the Record separator character is the same as the selected Field separator character. To enable this field and select a different character, uncheck the Same as Field separator checkbox.</p>
Same as Field separator	<p>Indicates if the tag separator is the same as the field separator. Uncheck this to select a different character as the tag separator.</p> <p>Note: By default, this checkbox is checked and the Tag separator field is disabled.</p>

Option Name	Description
Text qualifier	<p>The character used to surround text values in a delimited file.</p> <p>For example, this record uses double quotes (") as a text qualifier.</p> <pre>"7200 13TH ST" "MIAMI" "FL" "33144"</pre> <p>The characters available to define as text qualifiers are:</p> <ul style="list-style-type: none"> • Single quote (') • Double quote (") <p>If the file uses a different text qualifier, click the ellipses button to select another character as a text qualifier.</p>
Record separator	<p>Specifies the character used to separate records in line a sequential or delimited file. This field is not available if you check the Use default EOL check box.</p> <p>The record separator settings available are:</p> <p>Linux (U+000A) A line feed character separates the records. This is the standard record separator for Linux systems.</p> <p>Macintosh (U+000D) A carriage return character separates the records. This is the standard record separator for Macintosh systems.</p> <p>Windows (U+000D U+000A) A carriage return followed by a line feed separates the records. This is the standard record separator for Windows systems.</p> <p>If your file uses a different record separator, click the ellipses button to select another character as a record separator.</p>
Use default EOL	<p>Specifies that the file's record separator is the default end of line (EOL) character used on the operating system on which the Spectrum Technology Platform server is running.</p> <p>Do not select this option if the file uses an EOL character that is different from the default EOL character used on the server's operating system. For example, if the file uses a Windows EOL but the server is running on Linux, do not check this option. Instead, select the Windows option in the Record separator field.</p>

Fields Tab

The **Fields** tab controls which fields from the dataflow are included in the output file.

Option Name	Description
Add	<p>Click to add a field to the output.</p> <p>For information about constructing dataflow fields for use with Write to Variable Format File, see Writing Flat Data to a Variable Format File on page 437.</p>
Modify	<p>Click to modify the name of the tag. This button is only enabled when a tag is selected. If the Use fixed-width tags option is enabled on the File Properties tab, the tag width is automatically adjusted if you enter a longer tag name.</p> <p>Note: Using this button to modify the root tag name has the same effect as modifying the value of the Root tag name field on the File Properties tab.</p>
Remove	<p>Removes the selected field from the output. If you remove a list field all child fields are also removed. If you remove a child field, just the selected child field is removed from the list field.</p>
XX Remove All	<p>Removes all the fields from the output.</p>
Move Up/Move Down	<p>Reorders the selected field.</p>

Runtime Tab

Option Name	Description
File name	<p>This displays the file defined on the File Properties tab.</p>

Option Name	Description				
Generate multiple files	<p>Select this option to write records to different files instead of writing all records to one file. The file to which each record is written is specified in the record itself. Each record must contain a field that specifies either a file name or the full file path of the file to which you want the record written. For example, if you want to send the stock prices of different companies (of various groups) to all the clients separately, this feature writes the stock prices of different companies into separate files that may be sent to each of the clients, if you so wish. If you enable the Generate multiple file option you must specify an output file on either the Spectrum Technology Platform server or on an FTP server. If you want to write data to a file on an FTP server you must define a connection to the file server using Spectrum Management Console.</p> <p>Note: The records in the column you select in the File path field must be in sorted order. Use this feature when record contains either a file name or the full file path of the file.</p>				
File path field	<p>Selects the field that contains the path (either a file name or the full file path) of the file to which you want to write the record. Note that only the simple type elements mapped directly to a root tag will be listed in the File path field. This field is only enabled if you select the Generate multiple files.</p>				
Write Mode	<p>Specifies whether to add the dataflow's output to the end of the file or to delete the existing data in the file before writing the output:</p> <table border="0"> <tr> <td>Overwrite</td> <td>Replaces the existing data in the output file each time the dataflow runs.</td> </tr> <tr> <td>Append</td> <td>Adds the dataflow's output to the end of the file without erasing the file's existing data.</td> </tr> </table>	Overwrite	Replaces the existing data in the output file each time the dataflow runs.	Append	Adds the dataflow's output to the end of the file without erasing the file's existing data.
Overwrite	Replaces the existing data in the output file each time the dataflow runs.				
Append	Adds the dataflow's output to the end of the file without erasing the file's existing data.				

Writing Flat Data to a Variable Format File

In a Spectrum Technology Platform dataflow each record has the same fields. However, in a variable format file, not all records contain the same fields. In order to write flat data from a dataflow to a variable format file you need to break up each record in the dataflow, grouping the fields from each record into list fields corresponding to the record types you want to use for the variable format file. A list field is a collection of fields. For example, the fields FirstName, LastName, Gender, Address, and Phone could be grouped together into a list field called AccountOwner.

To write flat data to a variable format file, use an Aggregator stage to group fields into list fields corresponding to the record types you want to write to the variable format file. To do this:

1. Place an Aggregator stage in your dataflow anywhere upstream from the Write to Variable Format File stage.
2. Double-click the Aggregator stage to open its options window.
3. Select **Group by** then click **Add**.

- In the **Group By** field, select the field that contains a unique identifier that can be used to identify related data. This field's value should be unique across the records in the flat data. For example, an account number, a social security number, or a phone number.

Note: The field you select should be sorted. If it is not, use a Sorter stage to sort the records by the field.

- Click **OK**.
- Select **Output lists** then click **Add**.

Each output list will represent one record type in the variable format file.

- Select **New data type** and in the **Type name** field specify the type of information that will be contained in this data type. This will become a record type in the variable format file. For example, this data type will contain records related to account transactions, you could name the type "AccountTransaction".
- In the **Name** field, enter the name you want to give to this field. This may be the same name you specify in the **Type name** field.
- Click **OK**.
- Select the data type you just created and click **Add**.
- Leave the option **Existing field** selected and select one of the fields you want to include in this data type then click **OK**. Remember that this will become a record type in the variable format file. Repeat to add additional fields to this record type.
- Create additional output lists for each record type you want to have in the variable format file. When finished, click **OK** to close the Aggregator options.

The fields coming out of the Aggregator stage are now grouped into list fields that correspond to the record types you want to include in the variable format file output.

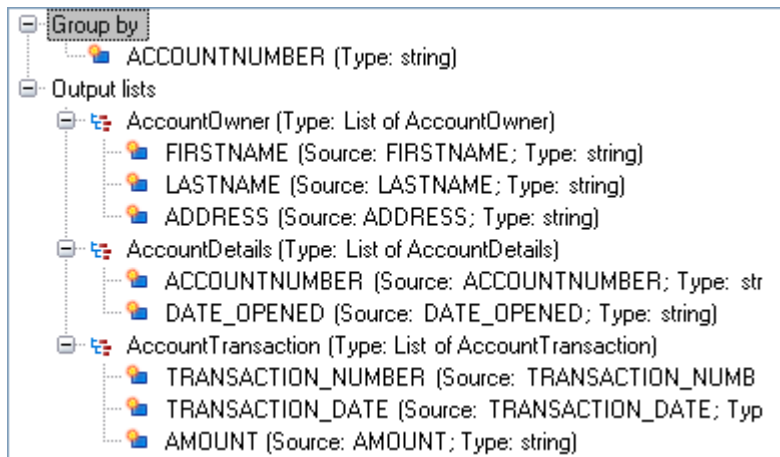
For example, given this flat data:

```
FIRSTNAME, LASTNAME, ADDRESS, ACCOUNTNUMBER, DATE_OPENED, TRANSACTION_NUMBER, TRANSACTION_DATE, AMOUNT
Joe, Smith, 100 Main St, CHK12904567, 12/2/2007, 1000567, 1/5/2012, 323.12
```

You would want to convert it to something like this in the variable format file:

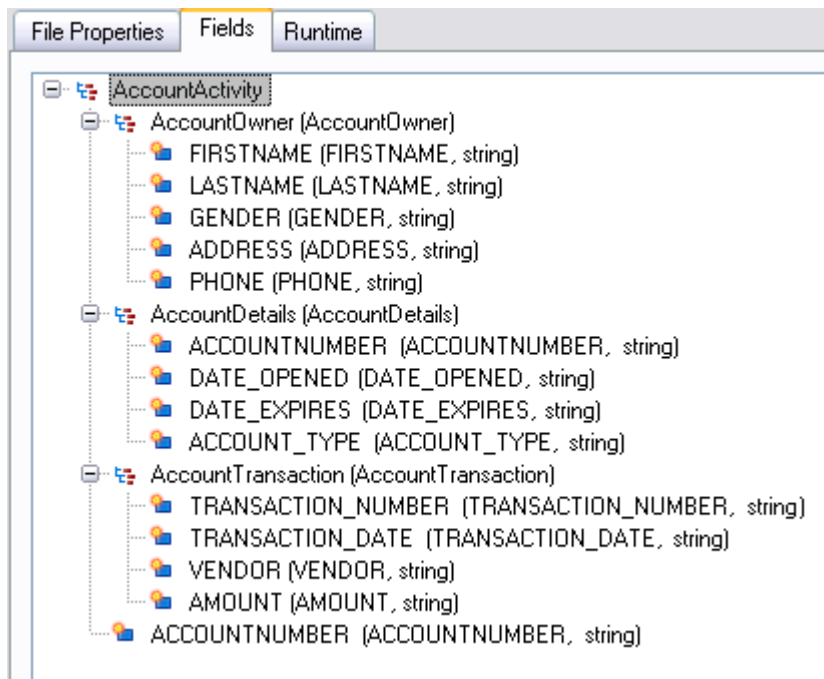
```
AccountOwner      Joe, Smith, 100 Main St
AccountInformation  CHK12904567, 12/2/2007
Transaction        1000567, 1/5/2012, 323.12
```

To accomplish this, you would create an Aggregator stage that is configured like this:



Tag Names in Variable Format Files

In a variable format file, each record in the output file has a tag which indicates the record type. In Write To Variable Format File, the field name is used as the tag name in the output file. For example, consider these fields:



These fields would be written to the file as follows. Note that in this example the account has two AccountTransaction records.

```
AccountOwner      Anne,Johnson,F,1202 Lake St,555-222-4932
AccountDetails    CHK238193875,1/21/2001,4/12/2012,CHK
AccountTransaction 1000232,3/5/2012,Blue Goose Grocery,132.11
AccountTransaction 1000232,3/8/2012,Trailway Bikes,540.00
```

Note: Only list fields containing simple fields such as strings are written to the output file. If a list field consists only of other list fields it is not written to the output file. In the above example, no record with an AccountActivity tag would be written to the output file because AccountActivity consists only of other list fields (AccountOwner, AccountDetails, and AccountTransaction).

Write to XML

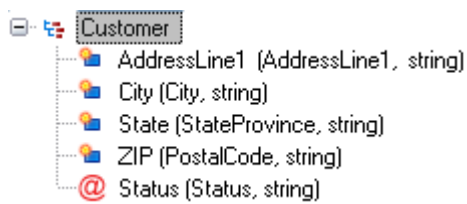
The **Write to XML** stage writes the output of a job or subflow to an XML file.

File Properties Tab

Field Name	Description
Data file	Specifies the path to the output XML file. Click the ellipses button (...) to locate the file you want. Note: If the Spectrum Technology Platform server is running on Linux, remember that file names and paths on these platforms are case sensitive.
Actual File	Displays the structure specified in the Fields tab. If you click an element and the file specified in the Data file field contains the element, a preview of the data will be displayed. Note that only data from simple elements can be displayed in the preview.
Export Schema	Click this button to save an XSD file that represents the schema shown in the Actual File view. The schema file is immediately saved to the location you specify.

Fields Tab

The **Fields** tab defines the fields you want to include in the output XML file. When you add fields, they are displayed in a tree structure. The tree displays the name of the element or attribute that will be written to the XML file. In parentheses following the element-attribute name is the name of the dataflow field followed by the data type, as in this example:



This indicates that four elements and one attribute will be written to the XML file. The attribute is indicated by the red "@" sign.

Note that the element State will contain the data from the field StateProvince and be string data. Likewise, the element ZIP will contain data from the PostalCode field and be string data. The XML file might look like this:

```
<XmlRoot>
  <Customer Status="0">
    <AddressLine1>7713 Mullen Dr</AddressLine1>
    <City>Austin</City>
    <State>TX</State>
    <ZIP>78757-1346</ZIP>
  </Customer>
  <Customer Status="0">
    <AddressLine1>1825B Kramer Ln</AddressLine1>
    <City>Austin</City>
    <State>TX</State>
    <ZIP>78758-4260</ZIP>
  </Customer>
</XmlRoot>
```

Note: The root element name (in this example <XmlRoot>) is specified on the **File Properties** tab.

The following table describes the options on the **Fields** tab.

Option Name	Description
Add	Adds a field to the output.

Option Name	Description
Modify	<p data-bbox="548 380 1289 407">Modifies how the field is written to XML. You can specify the following:</p> <p data-bbox="558 426 1406 512">Output type This option is available if you are modifying a simple field. It specifies whether the dataflow field should be written to an XML element or attribute.</p> <p data-bbox="737 531 1386 617">Element Select this to write the field's data to an XML element. Specify the element name you want to use in the Element name field.</p> <p data-bbox="737 636 1406 722">Attribute Writes the field's data to an attribute of the parent element. Specify the attribute name you want to use in the Attribute name field.</p> <p data-bbox="558 762 1419 848">Element name/Attribute name Specifies the name of the element or attribute to be written to the XML file. The default name is the dataflow field name.</p> <p data-bbox="558 867 1419 953">Change all children to This option is available if you are modifying a complex element. It specifies the type of XML you want the complex element to contain. One of the following:</p> <p data-bbox="737 972 1373 1100">No change The child types remain as they are currently defined, either element or attribute. You can specify the type for each field individually by selecting the field and clicking Modify.</p> <p data-bbox="737 1119 1386 1184">Elements All simple fields under the element are written as XML elements.</p> <p data-bbox="737 1203 1386 1268">Attributes All simple fields under the element are written as XML attributes.</p> <p data-bbox="558 1287 1419 1373">Namespace If you want to specify an XML namespace to use for the element or attribute, select it here. You can create namespaces on the Fields tab of the Write to XML stage.</p> <p data-bbox="558 1392 1419 1499">Include empty fields Check this box to include in the output file XML elements that have a null value or no data. If you do not check this box, empty elements will not be included in the output.</p> <p data-bbox="737 1518 1419 1604">For example, if you define an element named <code><City></code> but there is a record that does not have any data in the City field, the XML output will contain the following if you check Include empty fields:</p> <pre data-bbox="737 1623 1117 1650"><City xs:nil="true"></City></pre> <p data-bbox="737 1669 1419 1734">If you do not check this box the <code><City></code> element will not be written to the output file.</p> <p data-bbox="634 1753 1419 1875">Note: Dataflow field displays the field whose data will be written to the element or attribute. This is displayed so that if you change the element or attribute name to something different you can still see which field's data is contained in the element or attribute.</p>

Option Name	Description
Remove	Removes the selected field from the output. If you remove a list field all child fields are also removed. If you remove a child field, just the selected child field is removed from the list field.
Remove All	Removes all the fields from the output.
Move Up/Move Down	Reorders the selected field. Note that you cannot move simple elements into complex elements. If you want to modify the elements in a complex element, you must modify your dataflow's Aggregator stage to include the dataflow fields you want in the complex element. For more information, see Creating Complex XML from Flat Data on page 444.
Regenerate	Replaces the fields currently defined with the fields coming into Write to XML from the upstream channel.

Runtime Tab

Option Name	Description
Generate multiple files	Select this option to write records to different files instead of writing all records to one file. The file to which each record is written is specified in the record itself. Each record must contain a field that specifies either a file name or the full file path of the file to which you want the record written. For example, if you want to send the stock prices of different companies (of various groups) to all the clients separately, this feature writes the stock prices of different companies into separate files that may be sent to each of the clients, if you so wish. If you enable the Generate multiple file option you must specify an output file on either the Spectrum Technology Platform server or on an FTP server. If you want to write data to a file on an FTP server you must define a connection to the file server using Spectrum Management Console. Note: The records in the column you select in the File path field must be in sorted order. Use this feature when record contains either a file name or the full file path of the file.
File path field	Selects the field that contains the path (either a file name or the full file path) of the file to which you want to write the record. Note that only the simple type elements mapped directly to a root will be listed in the File path field . This field is only enabled if you select the Generate multiple files .

Option Name	Description
Generate schema at runtime	Select this option to generate an XSD at runtime and insert a <code>noNamespaceSchemaLocation</code> reference to schema in the XML file. The value of attribute <code>noNamespaceSchemaLocation</code> is XSD file name in the XML file. If you export the schema while editing a dataflow, there will be no reference to the XSD in the output XML file and the user would need to manually add in the reference to the XSD.
Schema path	Specifies the path to save the XSD file that contains the schema of the output XML file. Click the ellipses button (...) to browse to the file you want. The schema file is saved to the location you specify when you run the dataflow.

Using Namespaces in an XML Output File

Namespaces allow you to have duplicate element and attribute names in your output by assigning each element or attribute to an XML namespace.

1. In Spectrum Enterprise Designer, open the dataflow.
2. Double-click the Write to XML stage on the canvas.
3. Click the **Fields** tab.
4. Define one or more namespaces:
 - a) In the **Prefix** column, enter the prefix you want to use to associate an element or attribute with the namespace.
 - b) In the **Namespace** column, specify the URL of the namespace.
 - c) Repeat to define as many namespaces as you want to use for the output XML file.
5. Associate one or more elements or attributes to the namespace.
 - a) On the **Fields** tab, select the element or attribute you want to associate with a namespace then click **Modify**, or create a new element or attribute by clicking **Add**.
 - b) In the **Namespace** field, choose the namespace prefix you want to associate with the element or attribute.
 - c) Click **OK**.

Creating Complex XML from Flat Data

Dataflows often produce records containing flat fields which get written to XML as a simple XML elements. If you want to organize flat fields into complex XML elements to produce hierarchical data, you can do so using one or more Aggregator stages.

For example, given this flat data where the first line is a header record:

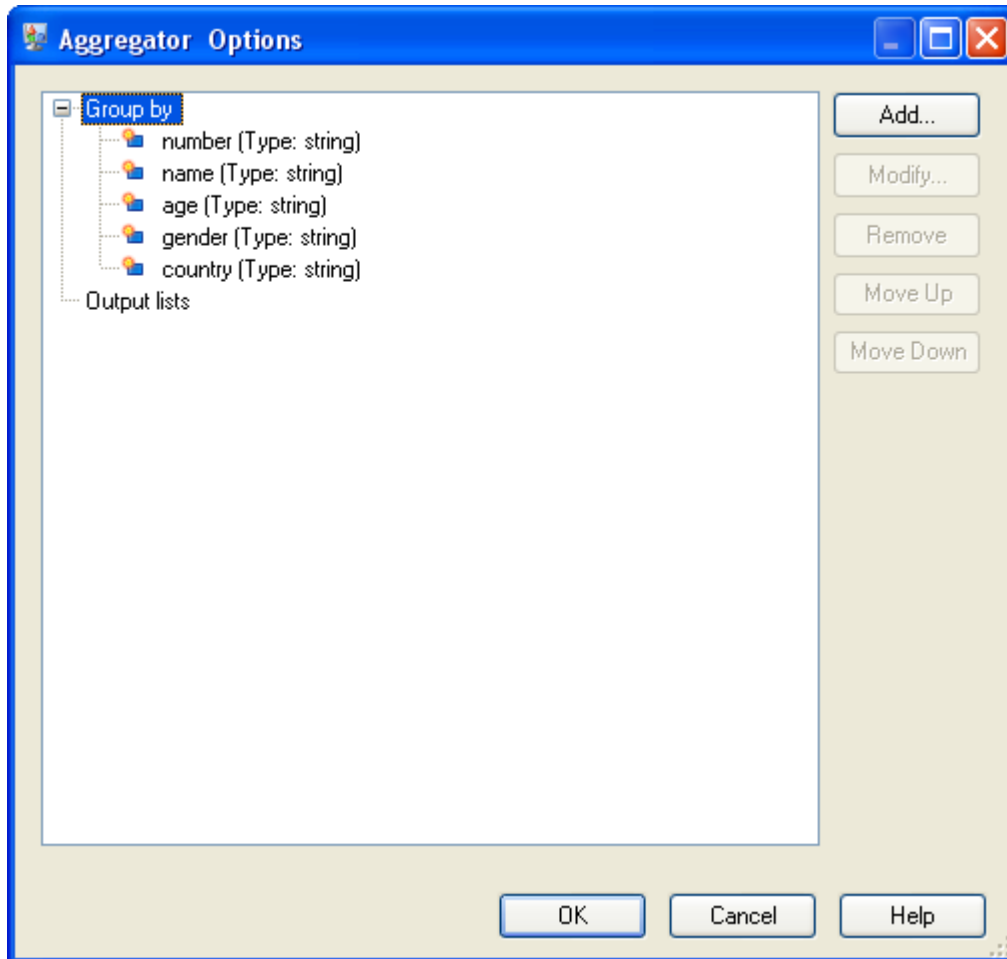
```
addressline1,age,city,country,gender,name,number,postalcode,stateprovince,type
1253 Summer St.,43,Boston,United States,M,Sam,019922,02110,MA,Savings
```

You might want to group the fields of data related to the address and fields related to the account into complex XML elements named `<Address>` and `<Account>` as shown here:

```
<CustomerRecord>
  <name>Sam</name>
  <age>43</age>
  <gender>M</gender>
  <country>United States</country>
  <Address>
    <addressline1>1253 Summer St.</addressline1>
    <city>Boston</city>
    <stateprovince>MA</stateprovince>
    <postalcode>02110</postalcode>
  </Address>
  <Account>
    <number>019922</number>
    <type>Savings</type>
  </Account>
</CustomerRecord>
```

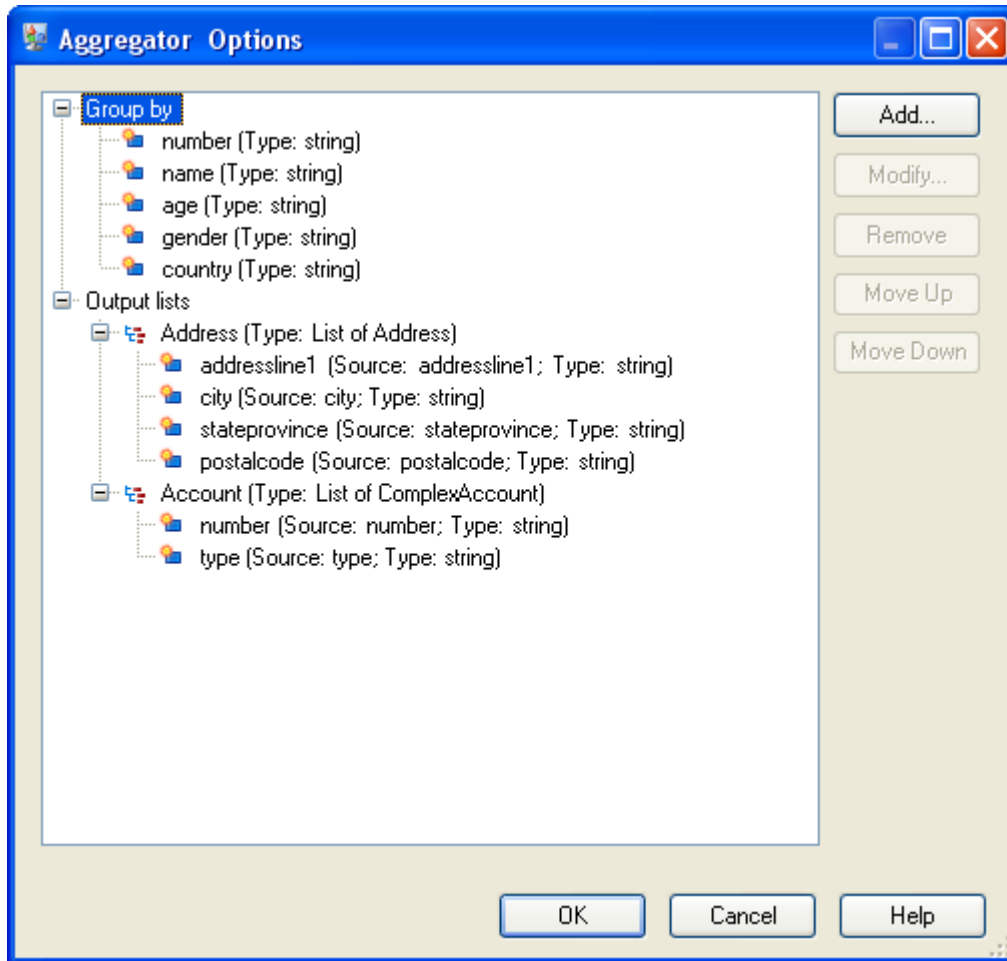
1. Add an Aggregator stage to the point in the dataflow where you want to construct complex elements.
2. Double-click the Aggregator stage to open the stage options.
3. Select **Group by** and click **Add**.
4. Select the field that contains a unique value for each record, such as an account number and click **OK**.
5. If there are other simple fields you want to pass through, select **Group by** and click **Add** again and add all the simple fields you want to include.

For example, in this case there are five simple fields that will be included in each record: number, name, age, gender, and country.



6. Select **Output lists** and click **Add**.
7. Select **New data type**. This will have the effect of defining a new complex element. Enter a description for the kind of data that this complex element will contain. For example, you could enter "Complex" since you are constructing a complex XML element. The data type name can be anything you want.
8. In the **Name** field, enter the name to use for the field. This will also be the name of the XML element.
9. Click **OK**.
10. Select the field you just created and click **Add**.
11. With **Existing field** selected, choose a field that you want to add as a child field to the complex element and click **OK**.
12. Repeat the previous two steps to add additional fields to the complex element.
13. Add additional complex fields as needed.

When you are finished, you should have an Aggregator stage that lists each simple and complex field you want to include in each record. For example:



14. Click **OK**.

Date and Number Patterns

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

Letter	Description	Example
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
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

Letter	Description	Example
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
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

Letter	Description	Example
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, 1.234×10^3 .

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 characters as literal characters in your number pattern's prefix or suffix, surround the special character with quotes.

Symbol	Description
0	<p>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:</p> <pre>0000</pre> <p>Would be:</p> <pre>0027</pre>

Symbol	Description
#	<p>Represents a digit but zeros are omitted. For example, the number twenty-seven when applied to this pattern:</p> <pre>####</pre> <p>Would be:</p> <pre>27</pre>
.	<p>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.</p>
-	<p>The negative sign used in the selected locale. For most locals this is the minus sign (-).</p>
,	<p>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.</p> <p>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:</p> <pre>#, ##, ###, ####</pre> <pre>#####, ####</pre> <pre>##, #####, ####</pre>
E	<p>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 451.</p>
;	<p>Separates positive and negative subpatterns. See Patterns for Negative Numbers on page 450.</p>
%	<p>Multiply the number by 100 and show the number as a percentage. For example, the number .35 when applied to this pattern:</p> <pre>##%</pre> <p>Would produce this result:</p> <pre>35%</pre>

Symbol	Description
¤	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, <pre>" '# '#"</pre> Formats 123 to: <pre>"#123"</pre> To create a single quote itself, use two in a row: <pre>"# o' 'clock"</pre>

Global Addressing Management Stages

Spectrum Global Address Validation

Spectrum Global Address Validation provides enhanced address standardization and validation. It is part of Spectrum Global Addressing Management.

Supported Countries

Spectrum Global Address Validation provides enhanced address standardization and validation for the following prioritized countries. The three-digit ISO country code is shown for each country. For a complete list of all ISO country codes, see [ISO Country Codes and Coder Support](#).

- Argentina (ARG)
- Australia (AUS)
- Austria (AUT)
- Belgium (BEL)
- Brazil (BRA)
- Canada (CAN)
- China (CHN)
- Czech Republic (CHZ)
- Denmark (DNK)
- Finland (FIN)
- France (FRA)
- Germany (DEU)

- Greece (GRC)
- India (IND)
- Ireland (IRL)
- Italy (ITA)
- Japan (JPN)
- Malaysia (MYS)
- Mexico (MEX)
- Netherlands (NLD)
- New Zealand (NZL)
- Norway (NOR)
- Poland (POL)
- Russia (RUS)
- Spain (ESP)
- Sweden (SWE)
- Switzerland (CHE)
- United Kingdom (GBR) (Includes POI information)
- United States (USA)

Spectrum Global Address Validation provides additional support for 130+ countries worldwide.

Using Spectrum Global Address Validation




After installing and deploying Spectrum Global Addressing Management, you can use Spectrum Global Address Validation:

- As a service from Management Console
- As a stage from Enterprise Designer

Using Spectrum Global Address Validation As a Service

To use Spectrum Global Address Validation as a service from Management Console:

1. Open the Management Console.
2. Under the Services tab, select **Global Addressing**.
3. From the list of services on the left side of the pane, select **Global Address Validation**.
4. On the **Database Resources** tab, select the Global Address Validation database resource to use.
 - a) Click **Enable International Processing** to perform global address processing and select your Global database.
 - b) Click **Enable US Processing** to perform United States (USA) address processing and select your US database.
5. Click **Save** to save your database selection.
6. Use the **Default Options** tab to define the default options for address processing. For information on the default options, see [Options](#) on page 455

- a) Click **Global Addressing Options** to define default options for global address processing. For information on the global addressing default option fields, see "Global Addressing Options" in **Options** on page 455.
 - b) Click **US Addressing Options** to define the default options for United States (USA) address processing. For information on the US addressing default option fields, see "US Addressing Options" in **Options** on page 455.
7. If you make changes to the global default options, click **Save** to save those changes. Any changes you make to the global default options are also applied to Global Address Validation in Enterprise Designer. If an Enterprise Designer job is open, you will need to refresh the job to pick up the changes.
 8. Use the **Output Options** tab to define the output options for address processing. For information on the output options fields, see "Output Options" in **Options** on page 455.
 9. Click the **Preview** tab.
 10. On the **Preview** tab, enter your input address elements in the appropriate fields. For information on the input fields, see **Input** on page 469.
 11. Click **Run Preview**.
 12. In Preview **Output Records**, on the right side of the pane, note that the results of the search have been placed in the appropriate output field. For information on the output fields, see **Output** on page 471.
 13. In the Preview tab, you can:
 - a) Use Add  to add additional input records for Run Preview processing.
 - b) Use Import  to import a number of input records for Run Preview processing.
 - c) Use Delete  to delete all records from the current Run Preview session.

Using Spectrum Global Address Validation As a Stage

You can use Spectrum Global Address Validation as a stage from Enterprise Designer to perform address validation as a batch process. For more information about creating a job using Global Address Validation as a stage, see:

- My First Dataflow (Job) in the Dataflow Designer Guide
- **Options**
- **Input**
- **Output**

Options

Global Address Validation uses the default options settings to define address validation processing.

Table 36: Global Addressing Options

Option Name	Country Support	Description
Default country	All	<p>The default country for address processing.</p> <p>To improve coding performance when your input addresses do not contain country information, set up an additional instance of the Global Address Validation stage as a preliminary stage to process and retrieve the country code for the input addresses.</p> <ol style="list-style-type: none"> Set up an additional instance of the Global Address Validation stage as a preliminary (first) stage in your dataflow. Give the preliminary stage a unique label. For example, "Identify Country". Specify "World" as the default country for the preliminary stage. The preliminary stage uses the available input address elements with additional data sources (available when you select "World" as the default country) to determine the country code. The "country-coded" output from the preliminary stage becomes the input for the next step in the dataflow. As the next step in the dataflow, the addresses are sent through a second Global Address Validation stage with the proper country code (retrieved in the preliminary stage) to validate the address to the street/house/premise level.

Global Addressing Options

Spectrum Global Address Validation uses the Global Addressing options settings to define address validation processing.

Table 37: Global Addressing Options

Option Name	Country Support	Description
Global Addressing Options	All except USA	The options specific to global address processing.
City fallback	All except USA	When a street level match cannot be made, use the input city to determine match candidates.
Postal fallback	All except USA	When a street level match cannot be made, use the input postal code to determine match candidates.

Matching Options

Global Address Validation uses the Matching options settings to define address validation processing.

Table 38: Matching Options

Option Name	Country Support	Description
Match mode	All	<p>Match modes determine the leniency used to make a match between the input address and the reference data. Select one of the following match modes based on the quality of your input and your desired output.</p> <p>Exact A very tight match. This restrictive mode generates the fewest match candidates. When using this mode, ensure that your input is very clean; free of misspellings and incomplete addresses.</p> <p>Relaxed A loose match. This mode generates the most match candidates and results in more multiple matches. Use this mode if you are not confident that your input is clean and free of misspellings and incomplete addresses.</p> <p style="text-align: center;">Note: For USA, the relaxed match mode is only allowed for the US Database Lookup.</p> <p>Custom A custom match. Allows you to define the matching criteria by selecting Custom Match Fields.</p>

Option Name	Country Support	Description
Maximum records to return		<p>The number of match candidates to return. Using a looser Match Mode setting such as "Relaxed" can result in the matching output including multiple match candidates.</p> <ul style="list-style-type: none"> • If an exact match is found, the single match candidate is returned. • If an exact match is not found and the maximum records to return option is set to a value other than 1, Global Address Validation returns the specified number of match candidate suggestions when match candidate suggestions are available. <p>For example, an exact match is not found. The maximum records to return is set to 3. Processing finds 8 match candidate suggestions. However, since maximum records to return is set to 3, only the three best match candidate suggestions are presented to the user to select the desired match candidate.</p> <p>In another example, an exact match is not found. The maximum records to return is set to 5. However, processing only finds two match candidate suggestions. In this example, only the two available match candidate suggestions are presented to the user to select the desired match candidate.</p> <p>Note: If you specify ON for the CASS Flag, CASS processing overrides the Maximum Records to Return to 1 (for the current processing run). Multiple candidate suggestions are not valid when CASS processing is turned ON.</p>
Prefer PO Box over street	CAN FRA GBR	Prefer candidates matching the input PO Box over matches to the input street. The default is disabled.
Prefer postal code over city	AUS	Prefer candidates matching the input postal code over matches to the input city. The default is disabled.

Custom Match Options

Global Address Validation uses the custom match options to set custom match criteria for determining match candidates. To enable these options, you must set the **Match Mode** to `Custom`. By default, these options are disabled.

Table 39: Custom Match Options

Option Name	Country Support	Description
Address number	All except USA	A match must be made to the input address number.
City	All except USA	A match must be made to the input address city.
City subdivision	All except USA	A match must be made to the input address city subdivision.
State/province	All except USA	A match must be made to the input address state or province.
State/province subdivision	All except USA	A match must be made to the input address state or province subdivision.
Street	All except USA	A match must be made to the input street name, type, and directional fields.
Postal code	All except USA	A match must be made to the input address postal code.

US Addressing Options

Spectrum Global Address Validation uses the US Addressing Options to define U.S. address processing.

Table 40: US Addressing Options

Option Name	Description
CASS flag	Process in a United States Postal Service (USPS) CASS-certified mode. Note: If you are performing Global Address Validation US Addressing processing outside of the physical United States, you must disable the Delivery Point Validation (DPV), LACSLink, SuiteLink, and Residential Delivery Indicator (RDI) options. You will not be able to run in a USPS CASS-certified mode.
Assign abbreviated city	Return abbreviated city names in the label lines.

Option Name	Description
Remove noise characters	Remove input noise characters (for example, unnecessary punctuation and blanks).
Return input firm	Return the input firm.
All street matching	Perform All Street Matching (ASM) processing. ASM applies additional matching logic to correct errors in street names to find a match. For example, when the first letter of a street is misspelled or missing on input, ASM searches all street names in a locality to find an input address. ASM provides the best address validation but may reduce performance. ASM processing is available for U.S. addresses only.
R777 deliverable	<p>Addresses with Carrier Route code R777 are phantom routes and are not eligible for street delivery. However, since these addresses are assigned a ZIP + 4 code by the USPS, these addresses are marked as deliverable. If you do not want the addresses with Carrier Route code R777 marked as deliverable, disable this option and the following actions are performed for the address:</p> <ul style="list-style-type: none"> • ZIP + 4 code is not assigned. • The address is not counted on the USPS Form 3553 (CASS Summary Report) • DPV Footnote Code R7 is returned.
Convert secondary to PMB	<p>Convert secondary information to Private Mail Box (PMB) under the following conditions:</p> <ul style="list-style-type: none"> • A secondary number is present in the returned ZIP + 4 address. • The secondary number does not DPV confirm. • The primary number (and/or other secondary number) confirms as a Commercial Mail Receiving Agency (CMRA). • The unconfirmed unit designator is not a pound sign (#). <p>Note: This processing only applies if the primary address codes to a CMRA.</p>
Save unit in separate field	Do not merge separate second unit and PMB information in the output address line. Save unit in a separate field.
Save PMB in separate field	Do not merge separate second unit and PMB information in the output address line. Save PMB in a separate field.

Option Name	Description
Return alias street	<p data-bbox="591 323 1422 380">Return alias street names in the label line. An alias street name is an alternate name for a street, maintained at the ranged ZIP + 4 Code level.</p> <ul data-bbox="591 407 1422 1016" style="list-style-type: none"> <li data-bbox="591 407 1422 520">• Return input alias or base. If the input address matches to an alias, return the alias. If the input address matches to a base address, but a preferred alias exists, return the preferred alias. If the input address matches to a base address and no Preferred alias exists, return the base address <li data-bbox="591 527 1422 583">• Return preferred alias or base. If a preferred alias exists, return the preferred alias. Otherwise, return the base street. <li data-bbox="591 590 1422 703">• Return preferred, abbreviated, input alias or base. Return the preferred alias. If no preferred alias exists, return the abbreviated alias. If no abbreviated or preferred alias exists, but some other alias type was input, return the input alias. If none of these scenarios apply, return the base street name. <li data-bbox="591 709 1422 802">• Return preferred, abbreviated alias, or base. If a preferred alias exists, return the preferred alias. If no preferred alias exists, return the Abbreviated alias. If neither exists, return the base street name. <li data-bbox="591 808 1422 921">• Return abbreviated, preferred, input alias or base. Return the abbreviated alias. If no abbreviated alias exists, return the preferred alias. If no abbreviated or preferred alias exists, but some other alias type was input, return the input alias. If no alias exists, return the base street name. <li data-bbox="591 928 1422 1016">• Return abbreviated, preferred, alias or base. Return the abbreviated alias. If no abbreviated alias exists, return the Preferred alias. If neither exists, return only the base street name

Option Name	Description
Dual address	<p>If the input file contains dual addresses (one a conventional address, a second containing a PO Box address), this field determines the order to use to process and match the addresses. If the selected address is valid, processing stops. If the selected address does not validate, processing attempts to code the secondary address.</p> <ul style="list-style-type: none"> • Above city and ZIP Code. The address line closest to the last line in the address is given the highest priority in the match process. Any address line above the last line is not used for matching. Default. • Line 1 is given preference. The first line in the dual address is given the highest priority in the match process. • Line 2 is given preference. The second line in the dual address is given the highest priority in the match process. • Conventional is given preference. The conventional address is given the highest priority in the match process. • PO Box is given preference. The PO Box is given the highest priority in the match process. • The first valid address is given preference. The first valid address (in the order Address line 1 and then Address line 2) is given the highest priority in the match process. <p>Note: When dual addresses are contained on a single line and the CASS flag is enabled, the USPS address type priority is used in the following order:</p> <ol style="list-style-type: none"> 1. PO Box 2. Firm 3. Highrise 4. Street 5. Rural Route 6. General Delivery
VeriMove data block	<p>The VeriMove data block option determines whether to return 250-bytes of additional information codes to the output file. This additional information can be used as input to Precisely's VeriMove product to comply with USPS® Move Update requirements.</p>

Additional Processing

Global Address Validation uses the default options settings to define additional processing options.

Table 41: Additional Processing Options

Option Name	Description
Delivery Point Validation (DPV)	<p>Perform Delivery Point Validation (DPV) processing. USPS CASS regulations require DPV processing. If you do not perform DPV processing, Global Address Validation does not generate a USPS Form 3553 (CASS Summary Report). The default is disabled.</p> <p>Note: When DPV is disabled, all DPV options are also disabled and display in a grayed out mode.</p> <p>Note: If you are performing Global Address Validation US Addressing processing outside of the physical United States, you must disable the Delivery Point Validation (DPV) option. You will not be able to run in a USPS CASS-certified mode.</p>
Tie Break	<p>The USPS allows DPV processing to be used as a tie breaker for matching inexact street records. If only one of the records in a tie is delivery point validated, a match is allowed to the inexact record. When processing results in an inexact match due to the input address directional, DPV processing can be used as a tie breaker if only one of the records is found to be delivery point validated and the delivery point validated record does not violate the cardinal direction rule. The default is enabled.</p> <p>USPS CASS regulations require DPV Tie Break processing to generate the USPS Form 3553 (USPS CASS Summary Report).</p>
Commercial Mail Receiving Agency (CMRA)	<p>Perform Commercial Mail Receiving Agency (CMRA) processing. A private company offering mailbox rental services to individuals and businesses is a Commercial Mail Receiving Agency (CMRA). The default is disabled.</p>
PO Box as Street Address (PBSA)	<p>Perform DPV P.O. Box Street Address (PBSA) processing. A PBSA address is a street address that represents a USPS P.O. Box. Use the PBSA table to identify PBSA addresses. Return the PBSA result to the output. The default is disabled.</p>
No-Stat	<p>Perform DPV No-Stat processing. Use the No-Stat table to identify deliveries that are not valid for Computerized Delivery Sequence (CDS) pre-processing. Return the proper No-Stat code to the output. The default is disabled.</p>
Vacant	<p>Perform DPV Vacant Table processing. Use the Vacant Table to identify delivery addresses that have been active in the past but, according to USPS data, have not been occupied within the last 90 days. Return the proper Vacant code to the output. The default is disabled.</p>

Option Name	Description
Door Not Accessible (DNA)	Perform DPV Door Not Accessible (DNA) processing. Use the DNA Table to identify delivery addresses where carriers cannot knock on the door for mail delivery or where carriers cannot physically access a residence/building such as rural/highway contract route (HCR), long driveway, or gated community. Return the proper DNA code to the output. The default is disabled.
Throwback	Perform DPV P.O. Box Throwback processing. Use the P.O. Box Throwback Table to identify a delivery point that is a street address where mail is not delivered. Instead, delivery is made to the customer's P.O. Box address. Return the P.O. Box Throwback result to the output. The default is disabled.
No Secure Location (NSL)	Perform DPV No Secure Location (NSL) processing. Use the NSL table to identify delivery locations that are not secure. For example, a carrier can access a door but cannot leave a package due to security concerns. The NSL designation alerts mailers to locations where businesses are closed on certain days and locations without mail receptacles (for example, a storefront). Return the NSL result to the output. The default is disabled.
Enhanced Line of Travel (eLOT)	Assign enhanced Line of Travel (eLOT) codes. The default is disabled.
Early Warning System (EWS)	Perform Early Warning System (EWS) processing. Perform Early Warning System (EWS) processing. New address information that is in use, but not yet available on the ZIP + 4 File, can be found as part of the USPS Early Warning System (EWS). The USPS requires all CASS-certified software to verify addresses that are not found in the current ZIP + 4 File against the USPS EWS File. If an address is found in the EWS File, the address is not matched to any similar addresses in the current ZIP + 4 File. Instead, the input address fails and is not coded until the ZIP + 4 File is updated with the correct address from the USPS EWS File. The default is disabled.
Residential Delivery Indicator (RDI)	Perform Residential Delivery Indicator (RDI) processing. The default is disabled. Note: If you are performing Global Address Validation US Addressing processing outside of the physical United States, you must disable the Residential Delivery Indicator (RDI) option. You will not be able to run in a USPS CASS-certified mode.
LACSLink	Perform LACSLink (Locatable Address Conversion System) processing. USPS CASS regulations require LACSLink processing. If you do not perform LACSLink processing, Global Address Validation does not generate a USPS Form 3553 (CASS Summary Report). The default is disabled. Note: If you are performing Global Address Validation US Addressing processing outside of the physical United States, you must disable the LACSLink option. You will not be able to run in a USPS CASS-certified mode.

Option Name	Description
SuiteLink	<p>Perform SuiteLink processing. USPS CASS regulations require SuiteLink processing. If you do not perform SuiteLink processing, Global Address Validation does not generate a USPS Form 3553 (CASS Summary Report). The default is disabled.</p> <p>Note: If you are performing Global Address Validation US Addressing processing outside of the physical United States, you must disable the SuiteLink option. You will not be able to run in a USPS CASS-certified mode.</p>
Return SuiteLink secondary	<p>Indicate how to return secondary information when SuiteLink secondary information is available.</p> <ul style="list-style-type: none"> • Both SuiteLink and input. Return both SuiteLink and input secondary information. Default. • SuiteLink only. Return SuiteLink secondary only. Do not return input secondary. • Input only. Return input secondary only. Do not return SuiteLink secondary. • None. Do not return SuiteLink secondary or input secondary.

CASS Mailer Information

The Mailer's name and address that displays on the USPS Form 3553 (CASS Summary Report). This information is required if you are running DPV and optional if you are not running DPV.

Table 42: CASS Mailer Information

Option Name	Description
Name	The Mailer's name displays in section D box 3 on the USPS Form 3553 (CASS Summary Report).
Address	Mailer's address. This information displays in section D box 3 on the USPS Form 3553 (CASS Summary Report).
Address2	Additional address line for Mailer's address displays in section D box 3 on the USPS Form 3553 (CASS Summary Report).
Address3	Additional address line for Mailer's address displays in section D box 3 on the USPS Form 3553 (CASS Summary Report).
Address4	Additional address line for Mailer's address displays in section D box 3 on the USPS Form 3553 (CASS Summary Report).

Option Name	Description
City, State, ZIP Code	The Mailer's city, state, and ZIP Code information displays in section D box 3 on the USPS Form 3553 (CASS Summary Report).

Multiple Address Line Options

Global Address Validation uses the multiple address line options settings to define the options specific to multiple address line processing. To specify the format of your returned lines address, select three options (one for each returned line and a special option).

Table 43: Multiple Address Line Options

Option Name	Description
Return Line 1	<p>Specify the format for returned line 1.</p> <ul style="list-style-type: none"> • Return the first valid line from the top. • Return the first firm line from top. • Return the first valid line above the city line. • Return the street address line above the city line. • Return the PO box or RR/HC line above the city line. • Return the best street or PO box line from top. If not found, the first firm or rural route line from top. When selecting this option for return line 1, processing selects from the top down in this order: <ol style="list-style-type: none"> 1. First PO box line or complete address. 2. First street line with a range but no suffix. 3. First Street line with a suffix but no range. 4. First rural route line. 5. First firm type.
Return Line 2	<p>Specify the format for returned line 2.</p> <ul style="list-style-type: none"> • Return a blank line. • Return the first valid line from the top. • Return the second valid line from the top. • Return the first firm line from top. • Return the second half of the combined address line if the first half is returned in line 1. • Return the first valid line above the city line. • Return the second valid line above city line. • Return the street address line above the city line. • Return the PO box or RR/HC line above the city line. • Return the first valid line above city line. If that criteria is not met resulting in a blank second line, return the topmost line not used in return line 2.

Option Name	Description
Special Options	<p>Specify special options for returned lines.</p> <ul style="list-style-type: none"> • Use all special options for returned lines. • Do not use any special options for returned lines. • Add a range of "0" to street lines without a range. For example, Main St becomes 0 Main St. • Separate combined address lines.
Return Line Order	<p>Specify the order for returning the standardized lines.</p> <ul style="list-style-type: none"> • 0 Firm line. • 1 PO Box address line. • 2 Address line with both a range and a suffix word. • 3 Address line with a range but no suffix, Address line with a suffix but no range. • 4 Rural route address line. • 5 Personal name, Firm name (w/o firm words), Unidentified. • 6 Apartment type line. • 7 Possible city line. • 8 City line. • 9 Ignore this line. • B Box address line following a rural route address line. • M Military address line. • N Best address line 1, best address line 2, city state, firm. URB and ZIP Code are returned in separate fields. The default is N. • R Rural route address line preceding a box line.
Do not concatenate lines beginning with #	Check Do not concatenate lines beginning with # if you do not want to concatenate lines beginning with a # to existing address lines.
Do not recognize periods (.) as valid characters	Check Do not recognize periods (.) as valid characters if you do not want to recognize periods as valid characters (periods should be removed before scanning address lines).
Do not concatenate dangling lines	Check Do not concatenate dangling lines if you do not want to concatenate one-word lines to address lines.
Do not merge secondary or PMB	Check Do not merge secondary or PMB if you do not want to merge a separated second unit and PMB information.
Do not identify FirmName	Check Do not identify FirmName if you do not want to perform firm name processing.

Option Name	Description
Do not identify CitySubdivision	Check Do not identify CitySubdivision if you do not want to perform Urbanization name (Puerto Rico) processing.
Do not identify LastLine	Check Do not identify LastLine if you do not want to perform last line (City, State, or ZIP Code) processing.

Log Level Options

Global Address Validation uses the log level options settings to define message logging.

Table 44: Log Level Options

Option Name	Description
Log level	The level of the messages to log. <ul style="list-style-type: none"> • No messages. • Critical messages • Error messages • Warning messages • Info messages • Debug messages

Output Options

Output Options define the elements to be returned by Spectrum Global Address Validation processing.

Table 45: Output Options

Option Name	Description
Parsed address	The parsed address elements (for example, Address Line 1, postal codes, and country). The meaning of some of these fields may vary by country. Do not select Parsed address when returning G/Z level matches.
Input address	Return the original input address.
Precision	Return a code describing the precision of the address match.
Country specific fields	Return country-specific output information.

Option Name	Description
Result Codes	Return result code information.
Casing	The format for the returned address: <ul style="list-style-type: none"> Mixed The output data is returned in a mixed case format. For example, 100 Main Street. Lower The output data is returned in an all lower case format. For example, 100 main street. Upper The output data is returned in an all upper case format. For example, 100 MAIN STREET. The default is Upper.

Input

Spectrum Global Address Validation uses an address as input. All addresses use this format regardless of the address's country. To obtain the best performance and address match, your input address lists should be as complete as possible, free of misspellings and incomplete addresses, and as close to postal authority standards as possible. Most postal authorities have websites that contain information about address standards for their particular country.

Note: The country name or two- or three- character country ISO code is optional. If you omit the country, Spectrum Global Address Validation returns the best available candidates for the **Default Country** selected on the **Default Options** tab. For a list of ISO codes, see [ISO Country Codes and Coder Support](#).

Table 46: Spectrum Global Address Validation Input

Field Name	Format	Description
FirmName	String	Company, firm name, or place name. For example, PRECISELY.
AddressLine1	String	The first address line. For example, 34 GLENVIEW ROAD MOUNT KURNING-GAI NSW 2080 . AddressLine1 can also contain a dual address (contains more than one mailable address). For example, the dual address PO BOX 3220 STN C 181 QUEEN STREET OTTAWA ON K1Y1E4 contains both a PO Box and a street address.
AddressLine2	String	The second address line (USA only).
AddressLine3	String	The third address line (USA only).
AddressLine4	String	The fourth address line (USA only).

Field Name	Format	Description
AddressLine5	String	The fifth address line (USA only).
AddressLine6	String	The sixth address line (USA only).
LastLine	String	<p>The last line of the address. For example, 34 GLENVIEW ROAD MOUNT KURNING-GAI NSW 2080.</p> <p>Note: Spectrum Global Address Validation only considers the LastLine information when individual components such as City and PostalCode are not provided.</p>
City	String	The city or town name. To produce the best match results, your input address should use the official city name.
CitySubdivision	String	<p>The name of one of the following depending on the country:</p> <ul style="list-style-type: none"> • Not used—AUS, AUT, BEL, CHE, DEU, DNK, FIN, FRA, IRL, MYS, NLD, NOR, POL, SWE • Dissemination Area and Enumeration Area (DA and EA)—CAN • Locality—BRA, GBR, GRC, ITA, ESP • Suburb—NZL • Urbanization name (Puerto Rico)—USA
StateProvince	String	<p>The name of the state or province depending on the country:</p> <ul style="list-style-type: none"> • Not used—BEL, CHE, DNK, IRL, NLD, NOR • Bundesland—DEU • Province—CAN • Province (voivodship)—POL • Region—AUT, ESP, FRA, GBR, GRC, NZL • Region (län)—FIN • Region (lan)—SWE • State—AUS, BRA, USA • State (negeri)—MYS

Field Name	Format	Description
StateProvinceSubdivision	String	The name of the state or province subdivision depending on the country: <ul style="list-style-type: none"> • Not used—AUT, BRA, CAN, FIN, GBR, MYS • Department—FRA • District—GRC • District (fylke/counties)—NOR • District (poviat)—POL • Kommun—SWE • Kreis—DEU • Local Government Authority (LGA)—AUS • Province—BEL, CHE, DNK, ESP, IRL, ITA, NLD • Region—NZL
PostalCode	String	The postal code in the appropriate format for the country.
Country	String	The country name of the two- or three-character ISO country code. This field is optional. If you omit the country, Spectrum Global Address Validation returns the best available candidates for the Default Country selected on the Default Options tab. For a list of ISO codes, see ISO Country Codes and Coder Support .

Output

Spectrum Global Address Validation output is determined by the output options you select.

Standard Address Output

Standard address output consists of address lines which correspond to how the address would appear on an address label. City, state or province, postal code, and other data are also included in the standard address output.

Table 47: Standard Address Output

Field Name	Format	Description
AdditionalInputData	String	Additional input data entered that was not used for matching.

Field Name	Format	Description
AddressBlock1-2	String	<p>The AddressBlock output fields contain a formatted version of the standardized or normalized address as it would be printed on a physical mailpiece. Global Address Validation formats the address into address blocks using postal authority standards. Each line of the address is returned in a separate address block field. There can be up to two address block output fields: AddressBlock1 and AddressBlock2.</p> <p>AddressBlock1 includes:</p> <ul style="list-style-type: none"> • PO Box—CAN, FRA, GBR • Firm Name • Unit Number • Unit Type • House Number • Street Name <p>AddressBlock2 includes:</p> <ul style="list-style-type: none"> • Locality • Town • Postal Code • County • State <p>For example, this input address:</p> <p>AddressLine1: 34 Glenview Road City: Mount Kurnung-Gai StateProvince: NSW PostalCode: 2080</p> <p>Results in this address block output:</p> <p>AddressBlock1: 34 Glenview Road AddressBlock2: Mount Kurnung-Gai NSW 2080</p> <p>In this example, the input address includes a PO Box (CAN, FRA, and GBR) and the "Prefer PO Box over street" option is selected:</p> <p>AddressLine1: 1 Great Lawn PO Box 10916 CM5 5AL</p> <p>Results in this address block output:</p> <p>AddressBlock1: PO Box 10916 AddressBlock2: ONGAR CM5 5AL</p>
AddressBlock3-10	String	<p>Additional AddressBlock fields for USA addresses. For USA addresses, starting from AddressBlock1 the data includes:</p> <ul style="list-style-type: none"> • Firm Name • URB name • Extra address line information • Required address line information • Last line

Field Name	Format	Description
AddressLine1	String	<p>The first address line.</p> <p>For example, 34 GLENVIEW ROAD MOUNT KURNING-GAI NSW 2080.</p> <p>In an example of a dual address, the address PO BOX 3220 STN C 181 QUEEN STREET OTTAWA ON K1Y1E4 contains both a PO Box and a street address.</p> <p>If the option "Prefer PO Box over street" is enabled, PO BOX 3220 STN C displays in this field.</p> <p>If the option "Prefer PO Box over street" is not enabled, 181 QUEEN STREET displays in this field.</p> <p>Global Address Validation supports PO Box matching for these countries:</p> <ul style="list-style-type: none"> • Canada (CAN) • France (FRA) • United Kingdom (GBR) • United States (USA)
AddressLine2	String	The second address line (USA only).
ApartmentLabel	String	The flat or unit type. For example, 39 Acacia Avenue Flat B .
ApartmentNumber	String	The flat or unit number. For example, 39 Acacia Avenue Flat B .
Building	String	The name of a building.
City	String	The city or town name. Your input address should use the official city name to produce the best match results.
City.Matched	String	<p>The status of the city match.</p> <p>True Matched on the city name.</p> <p>False Did not match on the city name.</p>
CitySubdivision	String	<p>The name of one of the following depending on the country:</p> <ul style="list-style-type: none"> • Not used—AUS, AUT, BEL, CHE, DEU, DNK, FIN, FRA, IRL, MYS, NLD, NOR, POL, SWE • Dissemination Area and Enumeration Area (DA and EA)—CAN • Locality—BRA, GBR, GRC, ITA, ESP • Suburb—NZL • Urbanization name (Puerto Rico)—USA

Field Name	Format	Description
CitySubdivision.Matched	String	The status of the match on city subdivision. True Matched on the city subdivision. False Did not match on the city subdivision.
Confidence	String	The level of confidence assigned to the address being returned. Range is from zero (0) to 100. Zero indicates failure. 100 indicates a very high level of confidence that the match results are correct.
Country	String	The country in the language or code specified in the Country format option. For a list of ISO codes, see ISO Country Codes and Coder Support .
Country specific fields	String	The country specific output information. To include the country specific output information in the output, check the Country specific fields output option.
FirmName	String	The name of a company.
Firmname.Matched	String	The status of the match on firm name. True Matched on the firm name. False Did not match on the firm name.
HouseNumber	String	The house number or PO Box number (CAN, FRA, UK). For example, 39 Acacia Avenue or PO Box 3220 .
Housenumber.Matched	String	The status of the match on house number. True Matched on the house number. False Did not match on the house number.
LeadingDirectional	String	The leading directional. For example, 123 E Main St Apt 3.
MatchOnAllStreetFields	String	The status of the match on all street fields. True Matched on all street fields. False Did not match on all street fields.

Field Name	Format	Description
MatchOnStreetDirectional	String	The status of the match on street directional. True Matched on the street directional. False Did not match on the street directional.
MatchScore	String	Reserved for future use.
MultimatchCount	String	If the address was matched to multiple candidate addresses in the reference data, this field contains the number of candidate matches found.
PostalCode	String	The postal code for the address. The format of the postal code varies by country.
PostalCode.AddOn	String	The second part of a postal code. This field is not used by most countries.
Postalcode.Matched	String	The status of the match on postal code. True Matched on the postal code. False Did not match on the postal code.
Principality	String	An area within a country. For example, England, Scotland, and Wales are principalities. This field will normally be blank.
ProcessedBy	String	The Spectrum Global Addressing Management stage name.
Result Code	String	The field-level result codes. Field-level result codes describe how each address element was processed. Field-level result codes are returned in the qualifier "Result". For example, the field-level result code for City is contained in City.Result. For a complete listing of result code output fields, see Result Codes.
StateProvince	String	The name of one of the state or province depending on the country: <ul style="list-style-type: none"> • Not used—BEL, CHE, DNK, IRL, NLD, NOR • Bundesland—DEU • Province—CAN • Province (voivodship)—POL • Region—AUT, ESP, FRA, GBR, GRC, NZL • Region (län)—FIN • Region (lan)—SWE • State—AUS, BRA, USA • State (negeri)—MYS

Field Name	Format	Description
StateProvince.Matched	String	The status of the match on state or province. True Matched on the state or province. False Did not match on the state or province.
StateProvinceSubdivision	String	The name of the state or province subdivision depending on the country. <ul style="list-style-type: none"> • Not used—AUT, BRA, CAN, FIN, GBR, MYS • County—USA • Department—FRA • District—GRC • District (fylke/counties)—NOR • District (powiat)—POL • Kommun—SWE • Kreis—DEU • Local Government Authority (LGA)—AUS • Province—BEL, CHE, DNK, ESP, IRL, ITA, NLD • Region—NZL
StateProvinceSubdivision.Matched	String	The status of the match on state or province subdivision. True Matched on the state or province subdivision. False Did not match on the state or province subdivision.
StreetName	String	The name of street where the property is located or "PO Box" to indicate the input record matched to a PO Box. For example, 123 E Main St or PO Box 3220.
StreetName.Matched	String	The status of the match on street name. True Matched on the street name. False Did not match on the street name.
StreetType	String	The street type. For example, 123 E Main St Apt 3. In another example, 123 E Main Ave Apt 3. These are two entirely different entities. Using street types adds precision to your data.
StreetType.Matched	String	The status of the match on street type. True Matched on the street type. False Did not match on the street type.

Field Name	Format	Description
TrailingDirectional	String	The trailing directional. For example, 123 Pennsylvania Ave NW .
VendorCode	String	The vendor code. This field is only available if you select the output option " Country specific fields ".

Parsed Input

Spectrum Global Address Validation output can include the input address in parsed form. This type of output is referred to as "parsed input." Parsed input fields contain the address data that was used as input regardless of whether or not Spectrum Global Address Validation validated the address. This information is not available when the address is validated at the postal / city level. It is available when an address gets validated at the street level. To include parsed input fields in the output, select the **Parsed address** output option.

Table 48: Parsed Input

Field Name	Format	Description
DualAddressParsed.Input	String	<p>A dual address is an address that contains more than one mailable address. For example, an address that contains both a PO Box and a street address is considered a dual address. When a dual address line is entered as input, this field contains the address line that is not used for AddressBlock1.</p> <p>If a PO Box (CAN, FRA, and UK) and a street address are entered as input and the option "Prefer PO Box over street" is enabled, the PO Box number is returned in the AddressLine1 field and AddressBlock1 fields and the street address is returned in the DualAddressParsed.Input field.</p> <p>If a PO Box (CAN, FRA, and UK) and a street address are entered as input and the option "Prefer PO Box over street" is not enabled, the street address is returned in the AddressLine1 field and AddressBlock1 fields and the PO Box is returned in the DualAddressParsed.Input field.</p>
ParsedAddressLine1.Input	String	<p>The first address line passed on input.</p> <p>For some countries (CAN, FRA, and UK), when the "PO Box over street" option is activated, this field contains PO Box.</p>
ParsedApartmentLabel.Input	String	The unit designator passed on input.
ParsedApartmentNumber.Input	String	The unit number passed on input.

Field Name	Format	Description
ParsedCity.Input	String	The city/locality/suburb name passed on input.
ParsedCitySubdivision.Input	String	The urbanization name passed on input.
ParsedCountry.Input	String	The country passed on input.
ParsedHouseNumber.Input	String	The house number passed on input. For example, 123 E Main St Apt 3.
ParsedPlaceName.Input	String	The place or firm name passed on input.
ParsedPostCodeAddOn.Input	String	The second part of a postal code passed on input. This field is not used by most countries.
ParsedPostCodeBase.Input	String	The postal code passed on input. For some countries, this field contains the first part of the postal code and the ParsedPostCodeAddOn.Input contains the second part of the postal code.
ParsedPostStreetType.Input	String	The street type passed on input. For example, 123 E Main St Apt 3.
ParsedPreStreetType.Input	String	The predirectional type passed on input. For example, 123 E Main St Apt 3.
ParsedStateProvince.Input	String	The name of one of the state or province depending on the country passed on input.
ParsedStateProvinceSubdivision.Input	String	The subdivision passed on input.

Precision

Spectrum Global Address Validation output can include the precision code that describes the precision of the address match for the input address. To include the precision code in the output, select the **Precision** output option.

Note: The "Precision code counts" section on the Match Analysis Report only displays when you check the "Precision" output option.

Table 49: Precision

Field Name	Format	Description
PrecisionCode	String	

Field Name	Format	Description
		A code describing the precision of the address match.
		The matches in the Z category indicate that a match was made at the postal code level.
	Z1	Match to ZIP Code™ or postal code 1.
	Z2	Match to ZIP + 2 or partial match to postal code 2.
	Z3	Match to ZIP + 4® or postal code 2.
		The matches in the G category indicate that the record was matched to an area name.
	G1	Match to state/province (area name 1).
	G2	Match to country/region (area name 2).
	G3	Match to city/town (area name 3).
	G4	Match to suburb/village (area name 4).
		The matches in the B category indicate that the record was matched to a PO Box.
	B1	Matched to an unvalidated PO Box. Although there is enough information in the record to identify this as a PO Box, not enough information exists to determine whether the PO Box number is valid.
	B2	Matched to a validated PO Box.
		The matches in the S category indicate that the record was matched to a single address candidate.
	S0	Single match; however, no coordinates are available. This is a very rare occurrence. Parts of the address may have matched the source data.
	S1	Single match to a ZIP Code™ or postal code 1 level. This is the same quality match as a Z1 result.
	S2	Single match to a ZIP + 2 or partial match to postal code 2 level. This is the same quality match as a Z2 result.
	S3	Single match to a ZIP + 4® or postal code 2 level. This is the same quality match as a Z3 result.
	S4	Single match at the street level.
	S5	Single match to the street address. Because only the street segment data is available, the interpolation is not as accurate as an S7 return. The S5 code is followed by letters and dashes indicating match precision.
	S6	Single match to a point located at a ZIP centroid.
	S7	Single match to a street address that was interpolated between houses.

Field Name	Format	Description
	S8	Single match to the street address or house number.
	SC	Single match at the house-level that has been projected from the nearest segment.
	SG	Single match with point at the center of a locality (<code>areaName3</code>) or Locality level geocode derived from topographic feature. An SG result code is associated with GNAF Reliability Level 5 (locality or neighborhood) or with Level 6 (unique region). (Australia addresses only.)
	SL	Single match to a sublocality (block or sector) street level match. An SL result code also requires a match on other geographic input fields (city, district, or state). (India addresses only.)
	SX	Single match to a point located at a street intersection.

For S (street matched) precision codes, eight additional characters describe how closely the address matches an address in the database. The characters appear in the order shown.

For example, the result code `S5--N-SCZA` represents a single match that matched the street name, street suffix direction, town, and postal code. The dashes indicate that there was no match on house number, street prefix direction, or thoroughfare type. The match came from the Street Range Address database. This record would be matched at the street address level of the match candidate.

Field Name	Format	Description
	H	House number match.
	P	Street prefix (pre-directional). P is present if any of these conditions are satisfied: <ul style="list-style-type: none"> • The candidate pre-directional matches the input pre-directional. • The candidate post-directional matches the input pre-directional after pre- and post-directionals are swapped. • The input does not have a pre-directional.
	N	Street name match.
	T	Street/thoroughfare type match.
	S	Street suffix (post-directional). <ul style="list-style-type: none"> • The candidate post-directional matches the input post-directional. • The candidate pre-directional matches the input post-directional after pre- and post-directionals are swapped. • The input does not have a post-directional.
	C	City or town name.
	Z	Postal code match.
	A	Addressing dataset match.
	U	Custom user dictionary match.

Single Match 'S' Precision Codes

The following table shows the support for the **S** category precision codes by country. For more information on the 's' precision codes, see **Table 49: Precision** on page 479. These descriptions apply to the vast majority of the countries. The exceptions for Australia and Canada are described in the sections that follow this table.

A bullet "•" indicates the **S** code is supported. A blank cell indicates the **S** code is not supported.

Country Name	S8	S7	S6	S5	S4	S3	S2	S1	S0	SX	SC	SG	SL
Australia (AUS)	•	•		•	•				•			•	
Canada (CAN)	•	•		•	•	•		•	•		•		

Country Name	S8	S7	S6	S5	S4	S3	S2	S1	S0	SX	SC	SG	SL
Denmark (DNK)	•	•		•	•					•			
Germany (DEU)	•	•		•	•					•			
Great Britain (GBR)	•	•		•	•				•	•			
India (IND)	•				•								•
New Zealand (NZL)	•	•		•	•					•			
All other countries	•	•		•	•				•	•	•		

Australia — 'S' Precision Code Descriptions

The following table provides 'S' precision code descriptions for Australia.

Result Code	Description
S8	Street level geocoded candidates return a result code beginning with the letter S. The second character in the code indicates the positional accuracy of the resulting point for the geocoded record.
S8.....G	Single match, point located at either the single point associated with an address point candidate or at an address point candidate that shares the same house number. No interpolation is required.
S7	The S8.....G result code is used for single matches with GNAF Reliability levels of 1 or 2 (the highest level of GNAF Reliability).
S7	Single match, located at an interpolated point along the candidate's street segment. When the potential candidate is not an address point candidate and there are no exact house number matches among other address point candidates, the S7 result is returned using address point interpolation.

Result Code	Description
S7.....G	The S7.....G result code is used for single matches with GNAF Reliability level of 3.
S5	Single match, point located at a street address position.
S4	Single match, point located at the center of a shape point path (shape points define the shape of the street polyline).
S4.....G	The S4.....G result code is used for single matches with a GNAF Reliability level of 4 (associated with a unique road feature).
S0	Single match, however, no coordinates are available (this is a very rare occurrence).
SX	Single match with the point located at street intersection.
SC	Single match where the original point has been moved a specified distance (usually along a perpendicular line) toward or away from the associated street segment. This result code can be returned only when both a point geocoding dataset and a street segment geocoding dataset are available and when the centerline offset feature is used.
SG	Single match with point at the center of a locality (<code>areaName3</code>) or Locality level geocode derived from topographic feature. An SG result code is associated with GNAF Reliability Level 5 (locality or neighborhood) or with Level 6 (unique region).

Canada — 'S' Precision Code Descriptions

The following table provides 'S' precision code descriptions for Canada.

Result Code	Description
Street level geocoded candidates return a result code beginning with the letter S. The second character in the code indicates the positional accuracy of the resulting point for the geocoded record.	
S8	Single match, point located at either the single point associated with an address point candidate or at an address point candidate that shares the same house number. No interpolation is required.
S7	Single match, located at an interpolated point along the candidate's street segment. When the potential candidate is not an address point candidate and there are no exact house number matches among other address point candidates, the S7 result is returned using address point interpolation.

Result Code	Description
s5	Single match, point located at a street address position.
s4	Single match, point located at the center of a shape point path (shape points define the shape of the street polyline).
s3	Single match, point located at postal centroid of FSALDU.
s1	Single match, point located at postal centroid of FSA.
s0	Single match, however, no coordinates are available (this is a very rare occurrence).
sc	Single match where the original point has been moved a specified distance (usually along a perpendicular line) toward or away from the associated street segment. This result code can be returned only when both a point geocoding dataset and a street segment geocoding dataset are available and when the centerline offset feature is used.

Result Codes

The result codes provide information on how Spectrum Global Address Validation processed U.S. addresses.

Table 50: Result Codes

Field Name	Format	Description
ApartmentLabel.Result	String	<p>The result codes for the apartment designator (for example, STE or APT).</p> <ul style="list-style-type: none"> - The input field was empty and supposed to be empty. A Appended. The field was added to a blank input field. C Corrected. D Dropped. The field provided on input was removed. M Multiple. The input address matched multiple records in the postal database. Each matching record has a different value in this field. S Standardized. This option includes any standard abbreviations. V Validated. The data was confirmed correct and remained unchanged from input.

Field Name	Format	Description
ApartmentNumber.Result	String	<p>The result codes for the apartment number (for example, Apt 3).</p> <ul style="list-style-type: none"> - The input field was empty and supposed to be empty. A Appended. The field was added to a blank input field. C Corrected. D Dropped. The field provided on input was removed. M Multiple. The input address matched multiple records in the postal database. Each matching record has a different value in this field. S Standardized. This option includes any standard abbreviations. V Validated. The data was confirmed correct and remained unchanged from input.
City.Result	String	<p>The result codes for the validated city name.</p> <ul style="list-style-type: none"> - The input field was empty and supposed to be empty. A Appended. The field was added to a blank input field. C Corrected. D Dropped. The field provided on input was removed. M Multiple. The input address matched multiple records in the postal database. Each matching record has a different value in this field. S Standardized. This option includes any standard abbreviations. V Validated. The data was confirmed correct and remained unchanged from input.

Field Name	Format	Description
CitySubdivision.Result	String	<p>The result codes for the validated urbanization name. This is primarily used for Puerto Rico addresses.</p> <ul style="list-style-type: none"> - The input field was empty and supposed to be empty. A Appended. The field was added to a blank input field. C Corrected. D Dropped. The field provided on input was removed. M Multiple. The input address matched multiple records in the postal database. Each matching record has a different value in this field. S Standardized. This option includes any standard abbreviations. V Validated. The data was confirmed correct and remained unchanged from input.
FirmName.Result	String	<p>The result codes for the validated firm or company name.</p> <ul style="list-style-type: none"> - The input field was empty and supposed to be empty. A Appended. The field was added to a blank input field. C Corrected. D Dropped. The field provided on input was removed. M Multiple. The input address matched multiple records in the postal database. Each matching record has a different value in this field. S Standardized. This option includes any standard abbreviations. V Validated. The data was confirmed correct and remained unchanged from input.

Field Name	Format	Description
HouseNumber.Result	String	<p>The result codes for the House number (for example, 123 E Main St Apt 3).</p> <ul style="list-style-type: none"> - The input field was empty and supposed to be empty. A Appended. The field was added to a blank input field. C Corrected. D Dropped. The field provided on input was removed. M Multiple. The input address matched multiple records in the postal database. Each matching record has a different value in this field. S Standardized. This option includes any standard abbreviations. V Validated. The data was confirmed correct and remained unchanged from input.
LeadingDirectional.Result	String	<p>The result codes for the leading directional (for example, 123 E Main St Apt 3).</p> <ul style="list-style-type: none"> - The input field was empty and supposed to be empty. A Appended. The field was added to a blank input field. C Corrected. D Dropped. The field provided on input was removed. M Multiple. The input address matched multiple records in the postal database. Each matching record has a different value in this field. S Standardized. This option includes any standard abbreviations. V Validated. The data was confirmed correct and remained unchanged from input.
POBox.Result	String	<p>The result codes for the Post office box number.</p> <ul style="list-style-type: none"> Blank Not applicable. C Corrected. S Standardized. This option includes any standard abbreviations. V Validated. The data was confirmed correct and remained unchanged from input.

Field Name	Format	Description
PostalCode.Result	String	<p>The result codes for the postal code. For U.S. addresses, this is the ZIP Code.</p> <p>Blank Not applicable.</p> <p>A Appended. The field was added to a blank input field.</p> <p>C Corrected.</p> <p>D Dropped. The field provided on input was removed.</p> <p>M Multiple. The input address matched multiple records in the postal database. Each matching record has a different value in this field.</p> <p>S Standardized. This option includes any standard abbreviations.</p> <p>V Validated. The data was confirmed correct and remained unchanged from input.</p>
PostalCode.Source	String	<p>The result codes for the postal code source.</p> <p>Blank Not applicable.</p> <p>ZIPMOVE The ZIP Code™ in the input address was corrected because the USPS® redrew ZIP Code™ boundaries and the address is now in a different ZIP Code™.</p>
PostalCode.Type	String	<p>The result codes for the postal code type.</p> <p>Blank Not applicable.</p> <p>P Post office box only.</p> <p>U Unique ZIP Code™.</p> <p>M Military ZIP Code™.</p>
RRHC.Result	String	<p>The result codes for the rural route/highway contract indicator.</p> <p>Blank Not applicable.</p> <p>C Corrected.</p> <p>S Standardized. This option includes any standard abbreviations.</p> <p>V Validated. The data was confirmed correct and remained unchanged from input.</p>

Field Name	Format	Description
RRHC.Type	String	The result codes for the rural route/highway contract indicator. Blank Not applicable. HC Highway contract route. RR Rural route.
StateProvince.Result	String	The result codes for the state or province name. Blank Not applicable. A Appended. The field was added to a blank input field. C Corrected. D Dropped. The field provided on input was removed. M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. S Standardized. This option includes any standard abbreviations. V Validated. The data was confirmed correct and remained unchanged from input.
StreetName.Alternate.Type	String	The alternate street name type. Blank Not applicable. A Alias (other). B Base street. D Alternate street. P Preferred alias. X Abbreviated alias.

Field Name	Format	Description
StreetName.Result	String	<p>The result codes for the street name (for example, 123 E Main St Apt 3).</p> <p>Blank Not applicable.</p> <p>A Appended. The field was added to a blank input field.</p> <p>C Corrected.</p> <p>D Dropped. The field provided on input was removed.</p> <p>M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field.</p> <p>S Standardized. This option includes any standard abbreviations.</p> <p>V Validated. The data was confirmed correct and remained unchanged from input.</p>
StreetName.Type	String	<p>The street name type.</p> <p>Blank Not applicable.</p> <p>A Alias (other).</p> <p>B Base street.</p> <p>D Alternate street.</p> <p>P Preferred alias.</p> <p>X Abbreviated alias.</p>
StreetSuffix.Result	String	<p>The result codes for the street name (for example, 123 E Main St Apt 3).</p> <p>Blank Not applicable.</p> <p>A Appended. The field was added to a blank input field.</p> <p>C Corrected.</p> <p>D Dropped. The field provided on input was removed.</p> <p>M Multiple. The input address matched multiple records in the postal database. Each matching record has a different value in this field.</p> <p>S Standardized. This option includes any standard abbreviations.</p> <p>V Validated. The data was confirmed correct and remained unchanged from input.</p>

Field Name	Format	Description
TrailingDirectional.Result	String	The result codes for the trailing directional (for example, 123 Pennsylvania Ave NW). Blank Not applicable. A Appended. The field was added to a blank input field. C Corrected. D Dropped. The field provided on input was removed. M Multiple. The input address matched multiple records in the postal database. Each matching record has a different value in this field. S Standardized. This option includes any standard abbreviations. V Validated. The data was confirmed correct and remained unchanged from input.

Input Address

Spectrum Global Address Validation output can include the input address. To include the input address, select the **Input address** output option.

Table 51: Input Address

Field Name	Format	Description
AddressLine1.Input	String	The first address line passed on input.
AddressLine2.Input	String	The second address line passed on input (USA only).
City.Input	String	The city/locality/suburb name passed on input.
CitySubdivision.Input	String	The city/town subdivision passed on input.
LastLine.Input	String	The last line passed on input.
StateProvince.Input	String	The state/province passed on input.
StateProvinceSubdivision.Input	String	The state/province subdivision passed on input.

Field Name	Format	Description
PostalCode.Input	String	The postal code passed on input.
Country.Input	String	The country passed on input.
FirmName.Input	String	The firm name passed on input.

Country Specific Fields

Spectrum Global Address Validation output can include country specific fields. To include country specific fields in the output, select the **Country specific fields** output option.

Table 52: Australia (AUS) Country Specific Fields

Field Name	Format	Description
AUS.Parcel.ID	String	The GNAF parcel identifier.
AUS.Pid	String	The GNAF Persistent Identifier (GNAF PID) is a 14-character alphanumeric string that uniquely identifies each GNAF address. The PID is constructed from a combination of the major address fields of the GNAF Dictionary. For example, GAACT718519668 .
AUS.Principal.Pid	String	The Persistent Identifier of the principal address.
AUS.Address.Class	String	The GNAF address classification.
AUS.SA1	String	The GNAF Statistical Area Level 1 (SA1) identifier.
AUS.Level.Number	String	The number of a floor or level in a multi-story building. For example, Floor 2 , 17 Jones Street.

Table 53: Canada (CAN) Country Specific Fields

Field Name	Format	Description
CAN.BuildingType	String	Indicates whether a building is a commercial building or an apartment building. 1 Residential (apartment) 2 Commercial <blank> indicates not a building
CAN.Census.CD	String	The Census Division (CD) in which the address is located. For more information about Census Divisions, see: http://www12.statcan.ca/english/census01/Products/Reference/dict/geo008.htm on the Statistics Canada website.
CAN.Census.CMA	String	The Census Metropolitan Area (CMA) in which the address is located. For more information about Census Metropolitan Areas, see http://www12.statcan.ca/english/census01/Products/Reference/dict/geo009.htm on the Statistics Canada website.
CAN.Census.CSD	String	The Census Subdivision (CSD) in which the address is located. For more information about Census Subdivisions, see http://www12.statcan.ca/english/census01/Products/Reference/dict/geo012.htm on the Statistics Canada website.
CAN.Census.CT	String	The Census Tract (CT) in which the address is located. For more information about Census Tracts, see http://www12.statcan.ca/english/census01/Products/Reference/dict/geo013.htm on the Statistics Canada website.
CAN.Census.DA	String	The Dissemination Area (DA) in which the address is located. For more information about Dissemination Areas, see http://www12.statcan.ca/english/census01/Products/Reference/dict/geo021.htm on the Statistics Canada website.

Table 54: India (IND) Country Specific Fields

Field Name	Format	Description
IND.Is.Rural	String	Indicates whether an address is located in a rural region (village).

Field Name	Format	Description
IND.POI.Category	String	Point of interest category. This field describes the type of POI, such as a bank, ATM, or restaurant.
IND.ExtendedResultCode	String	Block information.
IND.SubLocality	String	The municipal division below locality level.

Table 55: Ireland (IRL) Country Specific Fields

Field Name	Format	Description
IRL.Eircode	String	<p>The Eircode for the address. The Eircode is a seven character alpha-numeric code made up of two parts.</p> <p>Routing key The first three characters define a principal post town span of delivery.</p> <p>Unique Identifier The last four characters uniquely identify each residential and business address.</p>

Table 56: Italy (ITA) Country Specific Field

Field Name	Format	Description
ITA.Historical.Postcode	String	The previous postal code for this address.

Table 57: Japan (JAP) Country Specific Fields

Field Name	Format	Description
JPN.BANCHI	String	The block number.
JPN.CHOMOKU	String	The city block number.
JPN.CHOAZA	String	The number for a group of city blocks.
JPN.GO	String	The house number.

Field Name	Format	Description
JPN.JUSHOCODE	String	A point ID that represents a unique address.

Table 58: Netherlands (NLD) Country Specific Field

Field Name	Format	Description
NLD.EXTENED_POST_CODE	String	The 6-digit postal code. The output contains a candidate's 6-digit postal code if one is available.

Table 59: New Zealand (NZL) Country Specific Fields

Field Name	Format	Description
NZL.Aliased.SUBURB	String	The New Zealand aliased suburb. An alternative to the officially-recognized suburb name.

Table 60: United Kingdom (GBR) Country Specific Fields

Field Name	Format	Description
GBR.DependentStreet.Name	String	The addresses in the United Kingdom may contain two street names: a main street name and dependent street name. Some addresses may not contain a street name at all.
GBR.Dependent.Locality	String	The dependent locality name. A dependent locality is a large village or district. For example, Wimbledon .
GBR.DoubleDependent.Locality	String	The double dependent locality name. A double dependent locality is a small village or subdistrict.
GBR.Historic.Postcode	String	If the input address contained an old postal code that has been replaced by a new postal code, this field contains the old postal code.
GBR.Aliased.Locality	String	A locality that is not part of the postal address.

Field Name	Format	Description
GBR.OSAPR	String	The Ordnance Survey Address Point reference (OSAPR). Each address has a unique OSAPR. OSAPRs are always 18 characters long and must start with the letters AP.
GBR.UPRN	String	The Unique Property Reference Number. The UPRN is a unique identifier that provides a persistent reference to a unique property, regardless of changes in the property name, status, subdivision, use (such as from single occupancy to multiple occupancy), or demolition of the property. All historic, alternative, and provisional addresses are recorded against the same UPRN. The UPRN field is not returned for Northern Ireland addresses.
GBR.RPC	String	The RPC identifies the positional accuracy of the candidate. The RPC describes the accuracy of the coordinates allocated to the address.

Table 61: United States (USA) Country Specific Fields

Field Name	Format	Description
USA.AbbreviatedCityName	String	The abbreviated city name.
USA.AddressLocation	String	The location where the address information was found. 01 Input address found on address line 1. 02 Input address found on address line 2. 03 Input address found on address lines 1 and 3. 04 Firm found on input address line 1. No address line found. 06 Input address found on address line 2. Firm found on address line 1. 08 Firm found on input address line 2. No address line found. 09 Input address found on address line 1. Firm found on address line 2. 80 Address line not found.
USA.AdvancedBarcode	String	The returned 14-digit barcode consisting of the beginning frame character, ZIP Code, ZIP + 4, delivery point, check digit, and end framing character.
USA.AltStreet	String	The returned alternate street name.

Field Name	Format	Description
USA.AltStreetType	String	The returned alternate street name type.
		B Base street.
		A Alias (other).
		D Alternate street.
		P Preferred alias.
		X Abbreviated alias.
USA.Apartment1	String	The first apartment (unit) field in the address. This field is used for output. This field will only be populated if the Save unit in separate field is selected. If selected, the data is not included on the AddressLineX field(s).
USA.Apartment2	String	The second apartment (unit) field in the address. This field is used for output. This field will only be populated if the Save unit in separate field is selected. If selected, the data is not included on the AddressLineX field(s).
USA.BCCheckDigit	String	The valid one-digit modulo check digit required for printing the correct barcode.
USA.CarrierRouteCode	String	The returned carrier route code.
USA.CASSAddressLine1	String	For successfully-coded addresses, the output label line Address Line 1 contains the coded address line information.
USA.CASSAddressLine2	String	For successfully-coded addresses, the output label line Address Line 2 contains the city/state/ZIP Code information.
USA.CASSCityName	String	The city name. The city name returned here is the city name mandated by USPS regulations. Variations of the city name (for example, full, abbreviated, and non-mailing) are returned in alternate fields.
USA.CongressionalDistrict	String	The returned congressional district.
USA.DefaultMatch	String	The returned default match.
		Y Carrier route, or ZIP + 4, or DPBC default values returned.
		Blank No default values returned.

Field Name	Format	Description
USA.DPV	String	The returned Delivery Point Validation (DPV) indicators.
		N The address is not a valid delivery point. The USPS cannot deliver mail to this address.
		Y The address is delivery point validated. Primary range and secondary range (when present) are valid. The USPS can deliver mail to this address.
		S This address contains a valid primary range. Secondary range is present but is not confirmed. The USPS can deliver mail to this address.
		D This address contains a valid primary range. Secondary range is missing. The USPS can deliver mail to this address.
USA.DPV.CMRA	String	The returned Commercial Mail Receiving Agents (CMRA) indicators.
		Y The address is a valid CMRA.
		N The address is a confirmed delivery point but is not a valid CMRA.
		Blank This field is blank if the address is not a confirmed delivery point.
USA.DPV.DNA	String	The DPV Door Not Accessible (DNA) Table status indicator. The DNA Table identifies delivery addresses where carriers cannot knock on the door for mail delivery or where carriers cannot physically access a residence/building such as rural/highway contact route (HCR), long driveway, or gated community.
		Y The address was found in the DPV DNA Table.
		N The address was not found in the DPV DNA Table.
		Blank The DPV DNA Table was not queried.
USA.DPV.FalsePositive	String	The DPV False Positive Flag.
		Y The address is not a confirmed delivery point and a positive response was received from the False Positive File.
		N The address is not a confirmed delivery point and a negative response is received from the False Positive File. This field is blank if the address is a confirmed delivery point.
		Blank The False/Positive Table was not queried.

Field Name	Format	Description
USA.DPV.Footnote	String	The returned DPV footnote code.
		A1 Input address did not match to the ZIP + 4 File.
		AA Input address matched to the ZIP + 4 File.
		BB Input address matched to DPV (all components).
		CC Input address primary number matched to DPV but secondary number did not match (present but invalid).
		F1 Input address matched to a military ZIP Code.
		G1 Input address matched to a General Delivery address.
		M1 Input address primary number missing.
		M3 Input address primary number is invalid.
		N1 Input address primary number matched to DPV but address is missing secondary number.
		P1 Input address missing PO Box, rural route, or highway contract number.
		P3 Input address PO Box, rural route, or highway contract number invalid.
		PB Input address is a PO Box Street Address (PBSA).
		R1 Input address matched to CMRA but secondary number is not present.
R7 Input address is a Carrier Route R777.		
RR Input address matched to CMRA.		
U1 Input address matched to a unique ZIP Code.		
USA.DPV.LeadingDirectional	String	The parsed street predirectional for the record creating the False Positive (Seed) Table violation.
USA.DPV.MatchedZIP	String	The parsed ZIP Code for the record creating the False Positive (Seed) Table violation.
USA.DPV.MatchedZIP4	String	The parsed Plus4 ZIP Code for the record creating the False Positive (Seed) Table violation.
USA.DPV.NoStat	String	The DPV No-Stat Table status.
		Y The address was found in the DPV No-Stat Table.
		N The address was not found in the DPV No-Stat Table.
		Blank The DPV No-Stat Table was not queried.

Field Name	Format	Description
USA.DPV.NSL	String	The DPV No Secure Location (NSL) Table status.
		Y The address was found in the in the DPV NSL Table.
		N The address was not found in the DPV NSL Table.
		Blank The DPV NSL Table was not queried.
USA.DPV.PBSAFound	String	The DPV PO Box Street Address (PBSA) Table status indicator.
		Y The address was found in the DPV PBSA Table.
		N The address was not found in the DPV PBSA Table.
		Blank The DPV PBSA Table was not queried.
USA.DPV.Range	String	The parsed street primary range for the record creating the False Positive (Seed) Table violation.
USA.DPV.SeedHit	String	The DPV False Positive (Seed) Table indicator.
		Y The address was found in the in the DPV False Positive (Seed) Table.
		N The address was not found in the DPV False Positive (Seed) Table.
USA.DPV.StreetName	String	The street name for the address creating the False Positive (Seed) Table violation.
USA.DPV.Suffix	String	The street suffix for the address creating the False Positive (Seed) Table violation.
USA.DPV.Throwback	String	The DPV P.O. Box Throwback Table indicator.
		Y The address was found in the in the DPV P.O. Box Throwback Table.
		N The address was not found in the DPV P.O. Box Throwback Table.
		Blank The DPV P.O. Box Throwback Table was not queried.
USA.DPV.TrailingDirectional	String	The street postdirectional for the address creating the False Positive (Seed) Table violation.
USA.DPV.UnitDesignator	String	The unit designator for the address creating the False Positive (Seed) Table violation.

Field Name	Format	Description
USA.DPV.UnitNumber	String	The unit number for the record creating the False Positive (Seed) Table violation.
USA.DPV.Vacant	String	The DPV Vacant Table indicator. Y The address was found in the DPV Vacant Table. N The address was not found in the DPV Vacant Table. Blank The DPV Vacant Table was not queried.
USA.DPV.ZIP4	String	The ZIP + 4 used for DPV processing.
USA.EWSFailure	String	The address was not matched because the address was found in the USPS Early Warning System (EWS) File.
USA.FIPSCountyNumber	String	The returned five-digit FIPS code. Positions 1 and 2 contain the state code. Positions 3 through 5 contain the county code. Used for output information only.
USA.FiveDigitBarcode	String	The returned five-digit barcode.
USA.FiveDigitScheme	String	The returned 5-digit combined ZIP Code.
USA.FullCityName	String	The full city name.
USA.LACS	String	The LACSLink status. L The address is eligible for LACSLink processing. Blank No LACSLink processing available.
USA.LACS.Indicator	String	The one-byte CASS Stage file LACS ^{Link} Indicator value. If you are not performing a CASS Stage test, this field can be ignored. If you are performing a CASS Stage test, use the value in this field to populate the stage record. Y Record found in the LACS ^{Link} False Positive (Seed) Table. N Record not found in the LACS ^{Link} False Positive (Seed) Table.
USA.LACS.PreLACSAddress	String	The input address before LACSLink processing.
USA.LACS.PreLACSLeadingDirectional	String	The input address street predirectional determined before LACSLink processing.

Field Name	Format	Description
USA.LACS.PreLACSMatchedZIP	String	The input address matched ZIP Code determined before LACSLink processing.
USA.LACS.PreLACSMatchedZIP4	String	The input address matched ZIP+4 Code determined before LACSLink processing.
USA.LACS.PreLACSRange	String	The input address street primary range determined before LACSLink processing.
USA.LACS.PreLACSStreetName	String	The input address street name determined before LACSLink processing.
USA.LACS.PreLACSSuffix	String	The input address street suffix determined before LACSLink processing.
USA.LACS.PreLACSTrailingDirectional	String	The input address street postdirectional determined before LACSLink processing.
USA.LACS.PreLACSUnitD	String	The input address unit designator determined before LACSLink processing.
USA.LACS.PreLACSUnitN	String	The input address unit number determined before LACSLink processing.
USA.LACS.ReturnCode	String	<p>The LACSLink return code.</p> <p>A LACS^{Link} processing successful. Record matched through LACS^{Link} processing.</p> <p>00 LACS^{Link} processing failed. No matching record found during LACS^{Link} processing.</p> <p>09 LACS^{Link} processing matched the input address to an older highrise default address. The address has been converted. However, rather than provide an imprecise address, LACS^{Link} processing does not provide a new address.</p> <p>14 LACS^{Link} processing failed. Match found during LACS^{Link} processing but conversion did not occur due to other USPS regulations.</p> <p>92 LACS^{Link} processing successful. Record matched through LACS^{Link} processing. Unit number dropped on input.</p>

Field Name	Format	Description
USA.LACS.SeedHit	String	Indicates if the address was found in the LACSLink False Positive (Seed) Table.
		Y The address was found in the LACS ^{Link} False Positive (Seed) Table.
		N The address was not found in the LACS ^{Link} False Positive (Seed) Table.
USA.LOTCode	String	The returned enhanced Line of Travel (eLOT) code. Used for output information only. If eLOT is unavailable, the default value is 0000D.
USA.LOTSequence	String	The last character of the eLOT code indicates eLOT sequence.
		A Ascending.
		D Descending.
USA.MatchLevel	String	The returned match level.
		F Firm record match.
		G General delivery match.
		H Highrise match.
		P PO Box match.
		R Rural route/highway contract match.
S Street level match.		
USA.NonMailingCityName	String	The non-mailing city name. A city name that is recognized by the USPS, but is not the preferred name for the ZIP Code. This is often a vanity name for the area.
USA.Parsed.AltPostDirectional	String	The parsed alternate post directional.
USA.Parsed.AltPreDirectional	String	The parsed alternate pre-directional.
USA.Parsed.AltRange	String	The parsed alternate range.
USA.Parsed.AltStreetName	String	The parsed alternate street name.
USA.Parsed.AltStreetSuffix	String	The parsed alternate suffix.
USA.Parsed.PMUnitDesignator	String	The parsed PMB or MSC designator.

Field Name	Format	Description
USA.Parsed.PMUnitNumber	String	The parsed PMB or MSC unit number.
USA.Parsed.PostDirectional	String	The parsed post-directional.
USA.Parsed.PreDirectional	String	The parsed pre-directional.
USA.Parsed.Range	String	The parsed primary range.
USA.Parsed.StreetName	String	The parsed street name.
USA.Parsed.StreetSuffix	String	The parsed street suffix.
USA.Parsed.Unit2Designator	String	The parsed second unit designator.
USA.Parsed.Unit2Number	String	The parsed second unit number.
USA.Parsed.UnitDesignator	String	The parsed unit designator.
USA.Parsed.UnitNumber	String	The parsed unit number.
USA.POBoxOnly	String	<p>The PO Box only delivery zone status indicator.</p> <p>Y The address ZIP Code is a PO Box only delivery zone.</p> <p>N The address ZIP Code is not a PO Box only delivery zone.</p> <p>Blank Unable to determine USPS ZIP Code for input address.</p>
USA.PostalBarcode	String	The returned delivery point barcode.
USA.PreferredCityName	String	<p>The preferred city name for ZIP Code.</p> <p>Note: For successfully-coded addresses, the USA.PreferredCityName and the USA.PreferredState fields are always populated.</p> <p>For non-coded addresses, the USA.PreferredCityName and the USA.PreferredState fields are populated in the following scenarios:</p> <ul style="list-style-type: none"> • ZIP Code only input (city not input, or not found). • Single ZIP Code city input (ZIP Code not input, or not found). • City/St/ZIP Code input and agree (ZIP Code is part of city). <p>For all other non-coded scenarios, the preferred fields are blank.</p>

Field Name	Format	Description
USA.PreferredState	String	<p>The preferred state abbreviation for the preferred city name.</p> <p>Note: For successfully-coded addresses, the USA.PreferredCityName and USA.PreferredState fields are always populated.</p> <p>For non-coded addresses, the USA.PreferredCityName and USA.PreferredState fields are populated in the following scenarios:</p> <ul style="list-style-type: none"> • ZIP Code only input (city not input, or not found). • Single ZIP Code city input (ZIP Code not input, or not found). • City/St/ZIP Code input and agree (ZIP Code is part of city). <p>For all other non-coded scenarios, the preferred fields are blank.</p>
USA.PrivateMailbox	String	The returned matched Private Mail Box (PMB) or Mail Stop Code (MSC).
USA.PrivateMailbox.Input	String	Reserved for future use.
USA.PrivateMailbox.Type	String	Reserved for future use.
USA.PrivateMailbox.Type.Input	String	Reserved for future use.
USA.RDI	String	<p>The returned Residential Delivery Indicator (RDI).</p> <p>Y The address is a residential delivery.</p> <p>N The address is a business delivery.</p> <p>Blank The address failed address lookup (did not return a ZIP+4), or RDI was not active.</p>
USA.SeasonalFlags	String	<p>The seasonal delivery indicator beginning with January. The seasonal indicators for the returned ZIP Code. These indicators identify, at the 5-digit ZIP Code level, the months in which seasonal addresses receive delivery. There are 12 monthly flags (January through December). A "Y" in one of the monthly slots indicates that seasonal addresses are delivered mail in the month indicated by that slot. This field is blank if there are no seasonal deliveries for the ZIP Code.</p> <p>Y Deliver mail in this month.</p> <p>N Do not deliver mail in this month.</p>

Field Name	Format	Description
USA.Status	String	The match status of the address.
		F The address failed to match.
		Blank The address was successfully matched.

Field Name	Format	Description
------------	--------	-------------

USA.Status.Code	String	
-----------------	--------	--

USA.Status.Description		
------------------------	--	--

Field Name	Format	Description
		The returned match status code.
	4101	No city, state, and ZIP Code in address.
	4102	No ZIP Code and no city name in address.
	4103	No ZIP Code and no state name in address.
	4104	Cannot match to Unique ZIP Code.
	4211	Invalid ZIP Code and no city name in address.
	4212	No ZIP Code and invalid city name in address.
	4213	Invalid ZIP Code and invalid city name in address.
	4301	No street name in input address.
	4399	Blank address record.
	4411	No primary street name found in the Global Address Validation database.
	4412	No primary names ranked with certainty.
	4421	Invalid range or house number.
	4422	Incorrect or missing directional.
	4423	Incorrect or missing suffix.
	4425	Incorrect or missing suffix and directional.
	4450	No range in input address.
	4451	Multiple component failure. An address component had multiple options causing the address to fail assignment.
	4460	EWS Failure. Address found on EWS table.
	4461	eLOT assignment has failed. Address coded successfully but eLOT code not assigned.
	4465	The address requires a firm. No firm was provided or the firm failed to match.
	4466	The address requires secondary addresses (there is no default street address). No secondary was provided or the secondary did not match.
	4467	The address coded but was flagged for ZIP Move processing. The address failed to meet the final ZIP Move criteria. ZIP Move processing requires an exact match of street, suffix, and directional (both pre and post).
	4500	Unable to code. If reason for inability to code cannot be determined, this error is issued.
	4600	Undeliverable address in the Global Address Validation database.

Field Name	Format	Description
		<p>Note: This status code can still be valid with a successfully matched address. The combination of the two codes indicates that this is a valid address but is not deliverable by USPS standards.</p>
		<p>4601 The address failed Delivery Point Validation (DPV) processing.</p>
		<p>4602 The address is flagged as Carrier Route R777 and is not eligible for street delivery. This status code is only generated if the option R777 Deliverable is not activated.</p>
		<p>4801 Address is locked and was not be processed.</p>
		<p>5101 Warning: Missing apt/suite number.</p>
		<p>5102 Warning: Apt/suite was input but is not valid.</p>
		<p>5103 Warning: Input firm name is missing or invalid.</p>
		<p>5104 Warning: Multiple firms returned for address.</p>
		<p>5105 Warning: PO Box number is invalid or unavailable.</p>
		<p>5106 Warning: Apt/Suite was input but not allowed for this address.</p>
		<p>5200 Information: Address bypassed counted correct in Process Unassign run.</p>
USA.StreetNameAliasType	String	<p>The type of alias street name. An alias street name is an alternate name for a street, maintained at the 5-digit ZIP Code level.</p> <p>Abbreviated The alias street name is an abbreviation of the street name. For example, HARTS-NM RD is an abbreviated alias for HARTSVILLE NEW MARLBORO RD.</p> <p>Changed The official street name changed and the alias street name reflects the new official street name. For example, if SHINGLE BROOK RD is changed to CANNING DR, then CANNING DR would be a changed alias type.</p> <p>Other The alias street name is made up of nicknames, other names, or common abbreviations for the street name.</p> <p>Preferred The alias street name is the USPS preferred alias street name.</p>

Field Name	Format	Description
USA.SuiteLink.Fidelity	String	The SuiteLink fidelity code.
		1 All words in the business name match.
		2 Acceptable match. One or more words in the business name did not match, but acceptance criteria was still met.
		3 Unacceptable match. One or more words in the business name did not match. Acceptance criteria was not met.
USA.SuiteLink.MatchCode	String	The SuiteLink match code.
		A SuiteLink match found.
		B No SuiteLink match found.
		C Business name normalized to a blank value.
		D ZIP + 4 Code not recognized as a high-rise default.
		E Suite ^{Link} database expired.
USA.SuiteLink.ReturnCode	String	The SuiteLink return code.
		A Successful SuiteLink match.
		00 Failed SuiteLink match.
USA.VeriMoveDataBlock	String	Reserved for future use.
USA.ZIPValid	String	The returned ZIP Code.
USA.ZIP4Valid	String	The returned ZIP + 4 Code.

Reports

Reports

Spectrum Global Address Validation can produce reports for batch processing. To create the report, in Enterprise Designer drag the report icon you want to the canvas. You do not need to draw a connector to the report. For instructions on how to use reports, see the *Spectrum Technology Platform Dataflow Designer's Guide*.

Match Analysis by Country

The Spectrum Global Address Validation Match Analysis by Country Report provides address matching summary statistics for each country processed in your job. For instructions on how to use reports, see the *Spectrum Technology Platform Dataflow Designer's Guide*.

Summary of Matched Elements

This section contains summary information on matched elements for each country.

Total country records	The total number of input records processed for the country listed.
House number matched	The number and the percentage of records that matched on the house number.
Street name matched	The number and the percentage of records that matched on the street name.
City name matched	The number and the percentage of records that matched on the city name.
Postal code matched	The number and the percentage of records that matched on the postal code.
State/Province matched	The number and the percentage of records that matched on the state/province.

Precision Code Counts

This section provides statistics on the number and the percentage of records in your job that matched for each precision code. The precision code describes the level of precision for each record's address match.

Note: The "Precision code counts" section on the Match Analysis Report only displays when you check the **Precision** output option.

Precision Code B Category

PO Box level candidates return a precision code beginning with the letter B. The number following the B in the precision code provides more detailed information on the accuracy of the match.

Precision Code B1	The number and percentage of records that matched to an unvalidated PO Box. Although there is enough information in the record to identify this as a PO Box, not enough information exists to determine whether the PO Box number is valid.
Precision Code B2	The number and percentage of records that matched to a validated PO Box.

Precision Code G Category

Geographic level candidates return a precision code beginning with the letter G. The number following the G in the precision code provides more detailed information on the accuracy of the match.

Precision Code G1	The number and percentage of records that match to state/province (area name 1).
--------------------------	--

Precision Code G2	The number and percentage of records that match to county/region (area name 2).
Precision Code G3	The number and percentage of records that match to city/town (area name 3).
Precision Code G4	The number and percentage of records that match to suburb/village (area name 4).

Precision Code S Category

Street level candidates return a precision code beginning with the letter S. The character following the S in the precision code provides more detailed information on the accuracy of the match.

Precision Code SC	The number and percentage of records that matched at the house-level that has been projected from the nearest segment.
Precision Code SG	The number and percentage of records that matched to a point at the center of a locality (area name 3) or locality level geocode. An SG result code is associated with GNAF Reliability Level 5 (locality or neighborhood) or with Level 6 (unique region). (Australia addresses only.)
Precision Code SL	The number and percentage of records that matched to a sublocality (block or sector) street level match. An SL result code also requires a match on other geographic input fields (city, district, or state). (India addresses only.)
Precision Code SX	The number and percentage of records that validated at a street intersection.
Precision Code S0	The number and percentage of records where parts of the address may have matched the source data.
Precision Code S1	The number and percentage of records that resulted in a single match to a ZIP Code™ or postal code 1 level. This is the same quality match as a Z1 result.
Precision Code S2	The number and percentage of records that resulted in a single match to a ZIP + 2 or partial match to postal code 2 level. This is the same quality match as a Z2 result.
Precision Code S3	The number and percentage of records that resulted in a single match to a ZIP + 4® or postal code 2 level. This is the same quality match as a Z3 result.
Precision Code S4	The number and percentage of records that matched at the street level.
Precision Code S5	The number and percentage of records that matched to the street address.
Precision Code S6	The number and percentage of records that matched to a single point located at a ZIP centroid.
Precision Code S7	The number and percentage of records that matched to a street address that was interpolated between houses.

Precision Code S8 The number and percentage of records that matched to the street address or house number.

For more information about the S precision code, see the Global Address Validation Output section.

Precision Code Z Category

The Z category indicates that a match was made at the postal code level. A postal code match is returned in either of these cases:

- You specified to match to postal code. The resulting match is located at the postal code with the following possible accuracy levels.
- There is no street level match and you specified to fall back to postal code.

Precision Code Z1 The number and percentage of records that match to the ZIP Code or postal code 1.

Precision Code Z2 The number and percentage of records that result in a ZIP + 2 or partial match to postal code 2.

Precision Code Z3 The number and percentage of records that match to ZIP + 4 or postal code 2.

Confidence Levels

This section provides a graphical representation of the percentage of records for each country that matched at different confidence levels. The confidence level assigned to a returned address ranges from zero (0) to 100. Zero indicates failure. 100 indicates a very high level of confidence that the match results are correct.

Note: The confidence levels are calculated as a percentage of matched records. Input records that fail (Status.Code=F) and do not match are not included in the confidence level calculations and are not included in the confidence level graph on the report.

Confidence level less than 40 The percentage of records that match at a confidence level less than 40 (low).

Confidence level 40-85 The percentage of records that match at a confidence level between 40 and 85 (medium).

Confidence level greater than 85 The percentage of records that match at a confidence level greater than 85 (high).

Summary of Matched Elements for: Unknown

This section of the report provides matching statistics on records for which the **input** country code was not recognized **and** the country was not determined via address match.

Note: This section only displays when the output from your job includes records that did not match on country.

Precision Code Counts for: Unknown

This section provides matching precision statistics on records for which the **input** country code was not recognized **and** the country was not determined via address match. The precision code describes the level of precision for each record's address match.

Note: The "Precision code counts" section on the Match Analysis Report only displays when you check the **Precision** output option.

Summary of Matched Elements for: All Countries

This section of the report provides matching statistics for all input addresses that matched on country for all countries combined.

Precision Code Counts for: All Countries

This section provides matching precision statistics for all input addresses that matched on country for all countries combined. The precision code describes the level of precision for each record's address match.

Note: The "Precision code counts" section on the Match Analysis Report only displays when you check the **Precision** output option.

Precision Code Definitions

This section provides a reference for the precision codes that display on the report. For more information on precision codes, see the Global Address Validation Output section.

Report Footer

The footer on each page displays the time the report was generated and the page number.

Address Matching Summary Report

The Address Matching Summary Report provides summary matching statistics for each country processed. For instructions on how to use reports, see the *Spectrum Technology Platform Dataflow Designer's Guide*.

Country

This section provides matching statistics for each country processed.

Country	This column lists each country processed in your job.
Matched Records	The number of records that were successfully matched.
Matched Records %	The percentage of records that were successfully matched.

Unmatched Records	The number of records that were not successfully matched.
Unmatched Records %	The percentage of records that were not successfully matched.
Total Records	The total number of records processed for the country.

All Records

This section lists the total matching results for all countries processed.

Matched Records	The total number of records that were successfully matched.
Matched Records %	The percentage of all records in your job that were successfully matched.
Unmatched Records	The total number of records that were not successfully matched.
Unmatched Records %	The percentage of all records in your job that were not successfully matched.
Total Records	The total number of matched and unmatched records processed in your job.

USPS Form 3553 (CASS Summary Report)

The United States Postal Service® (USPS) Form 3553 (CASS Summary Report) is a facsimile of the Postal Form 3553. Global Address Validation generates this form automatically when you are using a USPS Coding Accuracy Support System (CASS) certified configuration. The USPS requires this form to verify CASS certification.

The second page of the USPS Form 3553 (CASS Summary Report) provides detailed information for each field on the form. For additional information on the USPS Form 3553 (CASS Summary Report), see <http://about.usps.com/forms/ps3553.pdf>. For instructions on how to use reports, see the *Spectrum Technology Platform Dataflow Designer's Guide*.

Spectrum Global Type Ahead

Spectrum Global Type Ahead automatically suggests addresses as you type and immediately returns candidates based on your input. You can then select your candidate from the presented candidate list. Spectrum Global Type Ahead is part of Spectrum Global Addressing Management.

Global Type Ahead Features

Global Type Ahead provides the following capabilities.

- Select country (optional). See **Supported Countries** on page 517 for a list of available countries.
- Single and multiple line input search for street addresses
- Specify the number of candidates to display
- Candidates returned in order based on closest match
- Search Points of Interest (POI)

Note: POI and category/subcategory features are accessible only if available and you have licensed and installed POI data.

- Search can include both street addresses and POI
- Additional filtering by City, State/Province, or Postal Code is available
- Fuzzy Match capabilities

For both address searches and POI searches, candidates are displayed as you type. As you type more specific information, the results are refined to display more relevant candidates. Candidates include the full address and POI (if the candidate is a Point of Interest).

After installing and deploying Global Type Ahead, you can use the Management Console to explore all the capabilities and see candidate results.

A sample application for Global Type Ahead is available on the Spectrum landing page under Spectrum Data Quality.

Supported Countries

Global Type Ahead covers street addresses and Points of Interest (POI) for the following countries. The three-digit ISO country code is shown for each country. For a complete list of all ISO country codes, see [ISO Country Codes and Coder Support](#).

Note: The POI data must be purchased separately. POI and category/subcategory features are accessible only if available and you have licensed and installed POI data. Street address data is packaged with the Spectrum Global Addressing Management.

- Andorra (AND)
- Australia (AUS)
- Austria (AUT)
- Bahrain (BHR)
- Belgium (BEL)
- Brazil (BRA)
- Canada (CAN)
- Czech Republic (CZE)
- Denmark (DNK)
- Finland (FIN)
- France (FRA)
- Germany (DEU)
- Greece (GRC) (Does not include POI information)
- Hungary (HUN)
- Ireland (IRL)
- Italy (ITA)
- Japan (JPN)
- Kuwait (KWT)
- Liechtenstein (LIE)

- Luxembourg (LUX)
- Mexico (MEX)
- Netherlands (NLD)
- New Zealand (NZL) (Does not include POI information)
- Norway (NOR)
- Oman (OMN)
- Poland (POL)
- Portugal (PRT)
- Qatar (QAT)
- Russia (RUS)
- Saudi Arabia (SAU)
- Singapore (SGP)
- Slovakia (SVK)
- Slovenia (SVN)
- South Africa (ZAF)
- Spain (ESP)
- Sweden (SWE)
- Switzerland (CHE)
- Thailand (THA)
- Turkey (TUR)
- United Arab Emirates (ARE)
- United Kingdom (GBR)
- United States (USA)

Note: See the current Database Release Notes for more details of country coverage and data vintages.

Using Global Type Ahead

After installing and deploying the Spectrum Global Addressing Management, you can use Global Type Ahead:

- As a service from Management Console
- As a stage from Enterprise Designer
- As a sample web application
- As a Java Script Component that you can use as a simple interface to integrate Global Type Ahead (GTA) functionality into an existing web application.

Using Global Type Ahead As a Service

To use Global Type Ahead as a service from Management Console:

1. Open the Management Console.
2. Under the Services tab, select **Global Addressing**.

3. From the list of services on the left side of the pane, select **Global Type Ahead**.
4. On the **Database Resources** tab, specify the Global Type Ahead database resource to use.
5. Click **Save** to save your database selection.
6. Click the **Default Options** tab.
7. On the **Default Options** tab, specify the options you want. For information on the options, see [Options](#) on page 519.
8. If you make changes to the global default options, click **Save** to save those changes. Any changes you make to the global default options are also applied to Global Type Ahead in Enterprise Designer.
9. Click the **Preview** tab.
10. In the **AddressLine1** field, enter the complete first line of the address, typically including street and house number.
11. In the **Country** field, enter the country name or the two or three-character ISO country code. If you omit the country, Global Type Ahead returns the best available candidates for the Default Country selected on the Default Options tab. For a list of ISO codes, see [ISO Country Codes and Coder Support](#).
12. You can further filter results by providing a city, state/province, or postal code.
13. Click **Run Preview**.
14. In Preview **Output Records** on the right side of the pane, note that the results of the search have been placed in the appropriate output field. For information on the output fields, see [Output](#) on page 521.

Using Global Type Ahead As a Stage

You can use Global Type Ahead as a stage from Enterprise Designer to perform address validation as a batch process. For more information about creating a job using Global Type Ahead as a stage, see:

- My First Dataflow (Job) in the Dataflow Designer Guide
- [Options](#) on page 519
- [Input](#) on page 521
- [Output](#) on page 521

Options

Global Type Ahead uses the default options settings to define address retrieval processing.

Table 62: Global Type Ahead Options

Option Name	Country Support	Description
Database	All	The database to use for Global Type Ahead processing. Only databases that have been defined in the Database Resources panel in the Management Console are available.
Default Country	All	The default country for address match processing.
Search type	All	The Global Type Ahead search options: <ul style="list-style-type: none"> Address Search for street addresses. Point Of Interest Search for Points of Interest (POI). City Search for street addresses within a specific city. State Search for street addresses within a specific state. Postal Search for street addresses within a specific postal code.
Max candidates	All	The maximum number of search candidates returned. The maximum is 99. The default is 5.
Fuzzy match	All	Global Type Ahead implements algorithms that optimize the retrieval of addresses and POIs, even when the input spelling is incorrect or incomplete. These capabilities are referred to as Fuzzy Match, and are implemented through match setting constraints. <ul style="list-style-type: none"> None Fuzzy Match is disabled by default. Hard Match The hard match allows one character substitution, insertion, deletion, or transposition. Soft Match The soft match allows two character substitutions, insertions, deletions, or transpositions.
Match on address number	All	You can specify Match on address number to determine if a house number match is required to get a match. If this match restriction is checked, then returned candidates must match the input house number. By default, the Match on address number box is unchecked, which means that returned candidates do not have to match the input house number. If the input does not contain a house number, the Match on address number restriction has no effect.

Input

In an interactive environment, Global Type Ahead automatically suggests addresses as you type and immediately returns candidates based on your input. Global Type Ahead can also return Points of Interest (POI).

Note: The POI data must be purchased separately. POI and category/subcategory features are available only if you have licensed and installed POI data. Street address data is packaged with the Global Addressing Module.

Table 63: Global Type Ahead Input

Field Name	Format	Description
AddressLine1	String	The complete first line of the address, typically including street and house number.
City	String	The city or town name.
StateProvince	String	The name of one of the state or province depending on the country.
PostCode	String	The postal code for the address. The format of the postal code varies by country.
Country	String	The country name or the two or three-character ISO country code. If you omit the country, Global Type Ahead returns the best available candidates for the Default Country selected on the Default Options tab. For a list of ISO codes, see ISO Country Codes and Coder Support .

Output

Global Type Ahead output is determined by the output options you selected.

Returned candidates can be previewed in the Management Console. Candidates include the complete address elements that you expect to see in Spectrum Technology Platform candidates, such as AddressLine, Range, City, County, State, and Country.

Note: The Global Type Ahead stage currently supports Range and Range Units for the United Kingdom (GBR) and the United States (USA). For the United Kingdom (GBR), UK Royal Mail (RM) data is used. For the United States (USA), Master Location Data (MLD) is used.

Table 64: Global Type Ahead Output

Field Name	Format	Description
AddressLine1	String	The complete first line of the address, typically including street and house number.
AddressNumber.Match	String	The status of the match attempt on address number. Returns true to indicate that the input address number matched the candidate. If address number is not matched, this field is not returned.
City	String	The city or town name.
City.Match	String	The status of the match attempt on city. Returns true to indicate that the input city matched the candidate. If city is not matched, this field is not returned.
County	String	The county name.
Country	String	The country name.
FirmName	String	The name of a company.
FormattedAddress	String	The formatted address.
LastLine	String	The last line of the address. For example, 10 DOWNING STREET LONDON, SW1A 2AA.
Locality	String	The locality.
PostalCode	String	The postal code for the address. The format of the postal code varies by country.
RangeCount	String	The number of ranges for the candidate.
Ranges	String	Additional information for each range identified for the candidate. <ul style="list-style-type: none"> • Range—The range number. • UnitCount—The number of units for the range. • UnitsInfo—Provides information for the unit and the formatted unit address.
StateProvince	String	The name of one of the state or province depending on the country.

Field Name	Format	Description
StreetName.Match	String	The status of the match attempt on street name. Returns true to indicate that the input street name matched the candidate. If street name is not matched, this field is not returned.
Type	String	Returns 1 for a POI match. Returns 2 for a street address match. POI and category/subcategory features are available only if you have licensed and installed POI data. You must have a license for POI data to include it and be able to return candidates for the dictionary.

Spectrum Global Type Ahead Sample Web Application

The Spectrum Global Type Ahead stage is packaged with a sample web application that demonstrates the Spectrum Global Type Ahead features and functionality. Global Type Ahead automatically suggests addresses as you type and immediately returns candidates based on your input. You can then select your candidate from the presented candidate list.

Note: Before using the sample web application, add a Global Type Ahead database resource in Management Console and save the database resource in the Global Type Ahead Service.

To use the Global Type Ahead sample web application:

1. Be sure the Spectrum Technology Platform server is running.
2. Open a web browser and go to: `http://<servername>:<port>/globaltypeahead`. For example, if your server is named "myserver" and your server uses the default HTTP port 8080, you would go to: `http://myserver:8080/globaltypeahead`. You can also find the sample web application for Global Type Ahead on the Spectrum™ landing page under Spectrum Data Quality.

Note: This site is best viewed in Internet Explorer 8.0 or later, Chrome, or Mozilla Firefox.

3. When the login screen appears, enter your user name and password.
4. Press **OK**.
5. Select a database from the drop-down list.
6. Select a country from the drop-down list.
7. Select the maximum number of candidates to display as you type addresses.
8. Select the appropriate match type.
9. Type an address in the address field. Address candidates display as you type. As you type more specific address information, the results are refined to display more relevant candidates.
10. You can filter results by providing a city, state, or postal code.
11. Select from the list of suggested addresses.
12. The selected address displays in the Search Result box.
13. To search for another address, click **Reset** to clear the fields.

Global Type Ahead Java Script Component

The Global Type Ahead Java Script Component is a simple interface that you can use to integrate Global Type Ahead (GTA) functionality into an existing web application. Global Type Ahead suggests candidate addresses based on your typed input. As you type more specific information, the results are refined to display more relevant candidates. You can then select your candidate from the suggested candidate list.

Requirements

The Global Type Ahead Java Script Component:

- Uses an AngularJS (1.x) web interface
- Uses a single line interface in the HTML source for your existing web application
- Maintains the Global Type Ahead Java Script Component options in a separate file from the main code
- Places the main Global Type Ahead Java Script Component code in a sub-folder to be included in your HTML application
- Calls the Spectrum Technology Platform and Spectrum Global Addressing Management interfaces using one of the following authentication methods:
 - No authentication
 - Session
 - Client
 - Open token
- Uses the Spectrum Technology Platform as the host service provider (for example, localhost:8080) for Global Type Ahead and Global Address Validation
- Accepts any web server to drive the web pages

Integrating Global Type Ahead Into Your Web Application

To integrate the Global Type Ahead functionality into an existing web application, you will need to:

- Review **Requirements** on page 524 if you have not already done so.
- Insert a few lines of code into your existing application.
- Edit the configuration file.
- Include a folder that provides the interface logic from the existing web application to the Global Type Ahead APIs.

Installing the Global Type Ahead Java Script Component

To install the Global Type Ahead Java Script Component, follow these steps:

1. The Global Type Ahead Java Script Component is installed as part of the Global Addressing Module installation. Locate **globaltypeahead.war** in the **Spectrum/server/app/deploy** folder.
2. Open the **globaltypeahead.war** file.

3. Locate the **WebWidget** subfolder. This folder contains the required files for using the Java Script Component.

Configuring Spectrum Technology Platform to Use the Global Type Ahead Java Script Component

After installing the Global Type Ahead Java Script Component, you must configure Spectrum Technology Platform to use the Global Type Ahead Java Script Component for your web application. If you have not already done so, review **Requirements** on page 524.

Enabling CORS

If you want to use an external web site to call Spectrum Technology Platform, Cross-Origin Resource Sharing (CORS) must be enabled. CORS prevents unauthorized web applications from stealing services from a server like the Spectrum Technology Platform. For more information, see the section "Enabling CORS" in your *Spectrum Technology Platform Administration Guide*.

1. Go to the **server/conf** folder.
2. Modify the following options in the **spectrum.properties** file.
3. Set the `spectrum.jetty.cors.enabled` property to true. The default is false.
4. Add your web server host name along with the port number to the `spectrum.jetty.cors.allowedOrigins` option.

Note: CORS only requires secure web access from a localhost (for example, <https://localhost:82>).

Non-secure access is allowed from a named server as shown in the following example:

```
spectrum.jetty.cors.allowedOrigins=http://us-8qxy12.pbi.global.pvt:82.
```

In this example, **us-8qxy12.pbi.global.pvt** is the machine name. **The machine name is case sensitive** so it is recommended that you check how your browser treats what you type. You need to change that to your server (machine) name. The **:82** is the port on which the web server is running. You configure the port when setting up the web server. For more information, see **Web Server**.

Authentication

Before using the Global Type Ahead Java Script Component, you must configure authentication for web service requests to the Spectrum Technology Platform server. For more information on the Spectrum Technology Platform authentication processing, refer to your *Spectrum Technology Platform Web Services Guide*.

To configure the Global Type Ahead Java Script Component authentication properties, follow these steps:

1. Edit the **spectrum-container.properties** file in the `Spectrum/app/conf` folder.
 - a) For **REST**, in the **spectrum-container.properties** file, set the value for the **spectrum.security.authentication.webservice.enabled.REST** property as needed. For

example, set `spectrum.security.authentication.webservice.enabled.REST=true` to enable authentication for all REST services.

Setting the value to `FALSE` removes any authentication requirements from Spectrum Technology Platform (not recommended). It is recommended that the values of both SOAP and REST be kept in sync.

- b) For **SOAP**, in the `spectrum-container.properties` file, set the value for the **spectrum.security.authentication.webservice.enabled.SOAP** property as needed. For example, set `spectrum.security.authentication.webservice.enabled.SOAP=true` to enable authentication for all SOAP services.

Setting the value to `FALSE` removes any authentication requirements from Spectrum Technology Platform (not recommended). It is recommended that the values of both SOAP and REST be kept in sync.

2. Enable CORS authentication. In the **spectrum.properties** file, add “, Authorization” to the end of the **spectrum.jetty.cors.allowedHeaders** option. For example:

```
spectrum.jetty.cors.allowedHeaders=X-PINGOTHER, Origin,
X-Requested-With, Content-Type, Accept, Authorization
```

Configuring the Global Type Ahead Java Script Component

After installing the Global Type Ahead Java Script Component, you must configure the tool for your web application. If you have not already done so, review [Requirements](#) on page 524.

Customizing the Global Type Ahead Java Script Component

To customize the Global Type Ahead Java Script Component for your use, follow these steps:

1. Edit the **autoCompleteDemoApp.js** file in the your root folder.
2. In the **spectrumServerName** field, enter the name of the Spectrum Technology Platform server including the port.
3. In the **authentication** field, enter one of the following for the type of the Spectrum Technology Platform authorization required.
 - None
 - Session
 - Client
 - Token
 - Self-created authentication token
4. In the **defaultCountry** field, specify the default country. Enter the full country name. You should specify the country where most of the addresses in your data are located. For example, if most of your addresses are in the United Kingdom, specify the United Kingdom. If you omit the country when entering addresses, Global Type Ahead returns the best available candidates for the Default Country specified.

5. In the **fuzzy** field, enter the type of matching logic to use. Global Type Ahead implements algorithms that optimize the retrieval of addresses, even when the input spelling is incorrect or incomplete. These capabilities are referred to as Fuzzy Match, and are implemented through match setting constraints.
 - **None**—Fuzzy Match is disabled by default.
 - **Hard Match**—The hard match allows one character substitution, insertion, deletion, or transposition.
 - **Soft Match**—The soft match allows two character substitutions, insertions, deletions, or transpositions.

Note: The use of fuzzy matching is only available in the alternative tool display.
6. In the **maxCandidatesReturned** field, enter a number between 1 and 99 for the maximum number of search candidates to be returned. The maximum is 99. The default is 5.
7. In the **sessionTimeout** field, enter the timeout value for token authentication in minutes. The default is 30.

Configuring Global Type Ahead Java Script Component Processing

To define the Global Type Ahead Java Script Component processing for your use, follow these steps:

1. Edit the **index.html** file in the your root folder.
2. Line 6 defines the AngularJS version being used for the Global Type Ahead Java Script Component. To replace the default AngularJS version, enter the version to use.
3. Line 7 makes the source code available to the web page.
4. Line 9 identifies the customization file. For more information, see [Customizing the Global Type Ahead Java Script Component](#) on page 526.
5. Lines 11, 12, and 13 define the Cascading Style Sheets (CSS) that drive the index.html web page. To replace the default CSS, enter the CSS to use with the Global Type Ahead Java Script Component.
6. Line 34 calls the Global Type Ahead Java Script Component.

The web server and the Spectrum Technology Platform server do not need to be on the same physical machine or platform. For example, you could have the web server running on Linux accessing a Spectrum Technology Platform server running on Windows.
7. The module name (for example, ng-app) and the controller (for example, ng-controller) need to match in both **WebWidget/autoCompleteDemoApp.js** and the **index.html** file.
8. **WebWidget\pb-address-complete\address-complete.js**, the location of the **template.html** file, needs to be accurate based on your setup.

Alternative Global Type Ahead Java Script Component Processing

The Global Type Ahead Java Script Component installs with a default interface. This interface is located in the pb-address-complete folder. The pb-address-complete folder contains the Global Type Ahead Java Script Component code.

If you would like to view the alternative interface:

1. Locate the **pb-address-complete** folder.
2. Rename template.html to template1.html.
3. Rename template2.html to template.html.
4. This exposes the alternative interface including the Fuzzy Matching option described [Customizing the Global Type Ahead Java Script Component](#) on page 526.

Using the Global Type Ahead Java Script Component

To use the Global Type Ahead Java Script Component:

1. Start Spectrum and your web server.
2. Open a web browser and point to your web server. For example, if your server is named "myserver" and your server uses port 82, you would go to: `http://myserver:82`.
3. Begin typing your address in the **Address Search** fields. Potential candidates start displaying with the third character typed.
4. If you type in an address that is a high rise with secondary ranges (APT numbers), the type ahead display indicates how many secondary ranges (APTs) are available. Click on the type ahead display item to see the secondary ranges for that address.
5. Select the final address.
6. The address is validated (using Global Address Validation) and displays displayed in Search Result section at the bottom of the page. The result is also exposed as fields for the end user which can further be used (as per application requirement) in the customer application or page. The fields are:
 - **selectedRangeltem** - This field outputs the object of the selected high-rise address with the high rise and includes secondary ranges (if any).
 - **selectedResult** - This field outputs the object of the secondary range from the above selected high-rise address or a possible selected candidate from the list.
 - **selectedAddress** - This field outputs the address format of the selected address that also appears in the type ahead text box.
7. To filter the results for the address you are typing, before typing the address, change the country or specify a city, state, or postal code. As you type a city or postal code, you will be provided with a list of validated alternatives.

Technical Notes

For more information on web service authentication, see the section "Web Service Authentication" in your *Spectrum Technology Web Services Guide*.

For more information on enabling CORS, see the section "Enabling CORS" in your *Spectrum Technology Platform Administration Guide*

The following tags in the index.html drive the widget processing:

- The <Link> tag defines the CSS (Cascading Style Sheets) that drives the index.html web page. You can remove the tag to see the effects that the CSS file defines or use your own CSS.
- This tag defines the version of AngularJS that is being included:
 - `<script src="/js/angular.min.js"></script>`
- This tag makes the widget source code available to the web page:
 - `<script src="/pb-address-complete/address-complete.js"></script>`
- This tag includes the customization file (see above):
 - `<script type="text/javascript" src="autoCompleteDemoApp.js"></script>`
- This tag includes the widget in the html page:
 - `<pb-address-complete options="options" selected-address = "selectedAddress" city = "city" country = "country" on-select="onSelect(address)"></pb-address-complete>`
- The web server and the Spectrum server do not need to be on the same physical machine or platform. For example, you could have the web server running on Linux accessing a Spectrum Server running on Windows.

Spectrum Global Address Parser

Spectrum Global Address Parser splits postal address strings into their constituent elements, such as name of organization, city, locality, district, and post code, using the machine learning techniques. It is a part of **Spectrum Global Addressing Management**.

You can feed data to the Spectrum Global Address Parser in these two ways:

- Enter addresses one at a time using the **Management Console**
- Import a comma-separated file of addresses arranged in a single column and having a header such as *address* in the **Management Console** or alternatively use any data source stage in the **Enterprise Designer**

This example shows an input address string and the corresponding formatted output:

Input string: "Widget Ltd Unit 5 Hatfield Business Park Mosquito Way Hatfield Hertfordshire AL10 9UJ GBR"

Formatted Output Record:

Field Name	Formatted Output
OrganizationName	Widget LTD

Field Name	Formatted Output
PlaceName	HATFIELD BUSINESS PARK
Floor	UNIT 5
Street	MOSQUITO WAY
City	HATFIELD
County	HERTFORDSHIRE
PostCode	AL10 9UJ
Country	GBR
Confidence.Total	78.64

Features of Global Address Parser

Global Address Parser can:

- Split and format address strings into components with the help of models trained through machine learning.
- Parse addresses in Roman script and accept input addresses in Roman script. Some Greek alphabets are also supported.
- Currently support parsing for:
 - Australia
 - Canada
 - France
 - Germany
 - Spain
 - United Kingdom
 - United States
- Handle country-specific addressing conventions efficiently. Address components of different countries vary in many ways. For example, in German addresses, house number usually comes after the street name and the post code comes before the city. The module handles all these complexities efficiently and predicts the address components in accordance with the conventions of the specified country.
- Eliminate the need for reference address databases for parsing.

Standard Fields

This table describes global address parsing terms and standard fields.

Fields	Descriptions
Supported Countries	Australia, Canada, France, Germany, Spain, United Kingdom, United States.
Formats	Handles country-specific addressing conventions efficiently. Primarily, the address pattern supported is 3 box: point information followed by street information, and location information.
Australia (AUS) Format	OrganizationName > AddressNumber > Street > City > StateProvince > PostCode > Country
Canada (CAN) Format	OrganizationName > AddressNumber > Street > PostCode > City > StateProvince > Country
France (FRA) Format	OrganizationName > AddressNumber > Street > PostCode > City > StateProvince > Country
Germany (DEU) Format	OrganizationName > Floor > PlaceName > AddressNumber > Street > Neighbourhood > City/Suburb/County > PostCode > Country
Spain (ESP) Format	OrganizationName > Street > AddressNumber > PostCode > City > StateProvince > Country
United Kingdom (GBR) Format	OrganizationName > Floor > PlaceName > AddressNumber > Street > Neighbourhood > City/Suburb/County > PostCode > Country
United States (USA) Format	OrganizationName > AddressNumber > Street > City > StateProvince > PostCode > Country
Use Case	Global Address Parser can break up single line addresses into address elements that can be used as input to a validation engine.
Output	
Confidence.Total	This confidence value indicates how sure the engine is of the fields parsed. Parsing is done on the basis of pre-shipped ML models. The patterns created by the machine learning engine are assigned a confidence value. You can use the confidence value to decide how to use the output fields as input to the address validation engine or other processes.

Guidelines to Improve Prediction Accuracy

In order to get the most accurate prediction of address components, your input address strings should adhere to these patterns.

Guidelines for Australia Addresses

Avoid non-address components	Presence of non-address components in the input string might lead to wrong prediction. Remove such components before feeding the string for prediction.
Maintain a sequence in address components	<p>The address components should be placed in this order: OrganizationName > AddressNumber > Street > PostCode > City > StateProvince > Country.</p> <p>Example:</p> <ul style="list-style-type: none"> • Incorrect: <i>Level 5 176 Messines Ridge Rd Griffith College Mount Gravatt QLD 4122 Australia</i> • Correct: <i>Griffith College Level 5 176 Messines Ridge Rd Mount Gravatt QLD 4122 Australia</i>
Remove redundant address components	<p>The input address string should not have repeated address components, such as two different organization names or repetitive name of an organization in one string.</p> <p>Example: <i>Griffith College Level 5 176 Messines Ridge Rd Griffith College Mount Gravatt QLD 4122 Australia</i></p>
Do not have merged components in address strings	<p>Merged address components result in incorrect prediction.</p> <p>Example:</p> <ul style="list-style-type: none"> • Incorrect: <i>Griffith College Level-5-176 Messines Ridge Rd Mount Gravatt QLD 4122 Australia</i> • Correct: <i>Griffith College Level 5 176 Messines Ridge Rd Mount Gravatt QLD 4122 Australia</i>
Avoid addressee name in the string	<p>Addressee name in the string results in incorrect prediction for the Australia addresses.</p> <p>Example:</p> <ul style="list-style-type: none"> • Incorrect: <i>Alice Smith Griffith College Level 5 176 Messines Ridge Rd Mount Gravatt QLD 4122 Australia</i> • Correct: <i>Griffith College Level 5 176 Messines Ridge Rd Mount Gravatt QLD 4122 Australia</i>
Do not have bracketed "()" address component	Including any of your address components inside brackets "()" will leave it unparsed.

Example: *Griffith College (Level 5) 176 Messines Ridge Rd
Mount Gravatt QLD 4122 Australia*

Limitations for Australia Addresses

These are the limitations of the address parser for Australia addresses:

- PO Box addresses are not supported.
- Sentence specific addresses (for example, addresses containing "close to", "between", "nearby") are not supported.

Example: *Tourquay Road Close To Butcher Shop Hervey Bay QLD 4655 AUS*

- Addresses containing roads with "and" or "&" are not supported.

Example: *Corner Farrall Road and O'Connor Road Stratton 6056 AUS*

- Addresses with a complex street format (for example, extra street information like tower, park, and building) are not supported.

Example: *Wesfarmers Limited Level 14 Brookfield Place Tower 2 123 St Georges Terrace Perth 6000 AUS*

- Unit/street components in character format are not supported.

Example: *Ground Floor 46 Charlotte St Brisbane 4000 AUS*

- Avoid repeating words for Org, state, or country in addresses (for example, Australia or QLD).

Example: *DOF Subsea Australia Pty Ltd 5th FL 181 St Georges TCE Perth Western Australia 6000 AUS*

Guidelines for Canada Addresses

Avoid non-address components

Presence of non-address components in the input string might lead to wrong prediction. Remove such components before feeding the string for prediction.

Maintain a sequence in address components

The address components should be placed in this order:
OrganizationName > AddressNumber > Street > PostCode > City > StateProvince > Country.

Example:

- **Incorrect:** *127 ORR AVE L4L9K2 ON WOODBRIDGE CAN*
- **Correct:** *127 ORR AVE L4L9K2 WOODBRIDGE ON CAN*

Remove redundant address components

The input address string should not have repeated address components, such as two different organization names or repetitive name of an organization in one string.

Example: *Adlib Publishing Systems Inc Adlib Publishing Systems Inc
10 5100 South Service Rd Burlington ON Canada*

Do not have merged components in address strings

Merged address components result in incorrect prediction.

Example:

- **Incorrect:** *Adlib-Publishing-Systems-Inc-10 South Service Rd Burlington ON Canada*
- **Correct:** *Adlib Publishing Systems Inc-10 South Service Rd Burlington ON Canada*

Avoid addressee name in the string

Address name in the string results in incorrect prediction for the Canada addresses.

Example:

- **Incorrect:** *Mr. XXX Adlib Publishing Systems Inc 10 5100 South Service Rd Burlington ON Canada*
- **Correct:** *Adlib Publishing Systems Inc 10 5100 South Service Rd Burlington ON Canada*

Do not have bracketed "()" address component

Including any of your address components inside brackets "()" will leave it unparsed.

Example: *(Adlib Publishing Systems Inc) 10 5100 South Service Rd Burlington ON Canada*

Limitations for Canada Addresses

These are the limitations of the address parser for Canada addresses:

- Unit or Apartment information is not supported.
- French characters present in the address are not displayed correctly.

Guidelines for France Addresses

Avoid non-address components

Presence of non-address components in the input string might lead to wrong prediction. Remove such components before feeding the string for prediction.

Maintain a sequence in address components

The address components should be placed in this order:

OrganizationName > AddressNumber > Street > PostCode > City > StateProvince > Country.

Example:

- **Incorrect:** *Normandie Hôtel 236 Rue Denis Papin Barentin Seine-Maritime 76360 France*
- **Correct:** *Normandie Hôtel 236 Rue Denis Papin 76360 Barentin Seine-Maritime France*

Remove redundant address components

The input address string should not have repeated address components, such as two different organization names or repetitive name of an organization in one string.

Example: *School André Marie 351 Boulevard de Normandie 76360 André Marie Barentin France*

Do not have merged components in address strings

Merged address components result in incorrect prediction.

Example:

- **Incorrect:** *School-André-Marie 351 Boulevard de Normandie 76360 Barentin Seine-Maritime France*
- **Correct:** *School André Marie 351 Boulevard de Normandie 76360 Barentin Seine-Maritime France*

Avoid addressee name in the string

Address name in the string results in incorrect prediction for the France addresses.

Example:

- **Incorrect:** *Mr. XXXXX School André Marie 351 Boulevard de Normandie 76360 Barentin Seine-Maritime France*
- **Correct:** *School André Marie 351 Boulevard de Normandie 76360 Barentin Seine-Maritime France*

Do not have bracketed "()" address component

Including any of your address components inside brackets "()" will leave it unparsed.

Example: *School (André Marie) 351 Boulevard de Normandie 76360 Barentin Seine-Maritime France*

Limitations for France Addresses

These are the limitations of the address parser for France addresses:

- Streets including city name are not supported.

Example: *14 Rue de Maule 78870 Bailly France*

- The overseas regions of France are incorrectly parsed (for example, Martinique, Réunion, and Guadeloupe).

*Guidelines for German Addresses***Avoid non-address components**

Presence of non-address components in the input string might lead to wrong prediction. Remove such components before feeding the string for prediction.

Maintain a sequence in address components

The address components should be placed in this order:
OrganizationName > Floor > PlaceName > AddressNumber >

Street > Neighbourhood > City/Suburb/County > PostCode > Country.

Example:

- **Incorrect:** *3 Weseler Strasse 46514 Schermbeck DEU*
- **Correct:** *Weseler Strasse 3 46514 Schermbeck DEU*

Remove redundant address components The input address string should not have repeated address components, such as two different organization names or repetitive name of an organization in one string.

Example: *Weseler Strasse 3 Weseler Strasse 46514 Schermbeck DEU*

Ensure address number and street name are included Your address string needs to have address number and street name. Missing out these essential address components will impact the accuracy of the result.

Example:

- **Incorrect:** *46514 Schermbeck DEU*
- **Correct:** *Weseler Strasse 3 46514 Schermbeck DEU*

Do not have merged components in address strings Merged address components result in incorrect prediction.

Example:

- **Incorrect:** *Weseler-Strasse-3 46514 Schermbeck DEU*
- **Correct:** *Weseler Strasse 3 46514 Schermbeck DEU*

Avoid addressee name in the string Addressee name in the string results in incorrect prediction for the German addresses.

Example:

- **Incorrect:** *Mr John Doe Weseler Strasse 3 46514 Schermbeck DEU*
- **Correct:** *Weseler Strasse 3 46514 Schermbeck DEU*

Do not have bracketed "()" address component

Including any of your address components inside brackets "()" will leave it unparsed.

Example: *Weseler Strasse 3 46514 (Schermbeck) DEU*

Guidelines for Spain Addresses

Avoid non-address components Presence of non-address components in the input string might lead to wrong prediction. Remove such components before feeding the string for prediction.

Maintain a sequence in address components The address components should be placed in this order:
OrganizationName > Street > AddressNumber > PostCode > City > StateProvince > Country.

Example:

- **Incorrect:** *Calle San Fernando 4 University of Seville 41004 Sevilla Spain*
- **Correct:** *University of Seville Calle San Fernando 4 41004 Sevilla Spain*

Remove redundant address components

The input address string should not have repeated address components, such as two different organization names or repetitive name of an organization in one string.

Example: *University of Seville Calle San Fernando 4 University of Seville 41004 Sevilla Spain*

Do not have merged components in address strings

Merged address components result in incorrect prediction.

Example:

- **Incorrect:** *University of Seville Calle-San-Fernando 4 41004 Sevilla Spain*
- **Correct:** *University of Seville Calle San Fernando 4 41004 Sevilla Spain*

Avoid addressee name in the string

Addressee name in the string results in incorrect prediction for Spain addresses.

Example:

- **Incorrect:** *Francisco Rodríguez University of Seville Calle San Fernando 4 41004 Sevilla Spain*
- **Correct:** *University of Seville Calle San Fernando 4 41004 Sevilla Spain*

Do not have bracketed "()" address component

Including any of your address components inside brackets "()" will leave it unparsed.

Example: *University of Seville (Calle San Fernando) 4 41004 Sevilla Spain*

Limitations for Spain Addresses

These are the limitations of the address parser for Spain addresses:

- Addresses starting with an abbreviation (for example, PL., Av., BL., C.) addresses are not supported.
- Streets including landmark information (for example, At, Near, Between) are not supported.

Guidelines for United Kingdom Addresses

Avoid non-address components

Presence of non-address components in the input string might lead to wrong prediction. Remove such components before feeding the string for prediction.

Maintain a sequence in address components

The address components should be placed in this order:

OrganizationName > **Floor** > **PlaceName** > **AddressNumber** > **Street** > **Neighbourhood** > **City/Suburb/County** > **PostCode** > **Country**.

Example:

- **Incorrect:** *Widget Limited London Milenium street Unit 3 AB10 3DF GBR*
- **Correct:** *Widget Limited Unit 3 Milenium street London AB10 3DF GBR*

Remove redundant address components

The input address string should not have repeated address components, such as two different organization names or repetitive name of an organization in one string.

Example:

- **Incorrect:** *Widget Limited Widget Limited Unit 10 Logix Cyber Park 10 Manor Street London AB10 3DF GBR*

Follow single-token organization names with organization type

A single-token organization name should be followed by the *type* of the organization, such as *Ltd*, *Inc*, and *Reg*. In the example below, *Ardian* is a single-token organization name. In this case, the organization name is not followed by the type "Limited," and the results may be inaccurate.

Example:

- **Incorrect:** *Ardian Fourth Floor Channel House St Helier Je2 4UH GBR*
- **Correct:** *Ardian Limited Fourth Floor Channel House St Helier Je2 4UH GBR*

Limitations for United Kingdom Addresses

An address string of any of these kind is susceptible to getting inaccurately predicted by the address parser. Watch out for these in your address strings.

Presence of another address component as name of the organization

If the name of the organization includes any other address component, such as `Floor`, `Flat`, and `House`, the prediction accuracy may be affected.

Example: *Flat Seasons 632 Kings Road London Middlesex SW6 2DU GBR*

Organization name having numbers

If an organization name has numbers, it is susceptible to getting erroneously predicted.

Example: *123 Limited ABC Street AB10 3DF GBR*

Guidelines for United States Addresses

- | | |
|---|---|
| Avoid non-address components | Presence of non-address components in the input string might lead to wrong prediction. Remove such components before feeding the string for prediction. |
| Maintain a sequence in address components | The address components should be placed in this order:
OrganizationName > AddressNumber > Street > City > StateProvince > PostCode > Country.
Example:
<ul style="list-style-type: none"> • Incorrect: <i>2200 Western CT Widget USA Lisle IL 60532</i> • Correct: <i>Widget 2200 Western CT Lisle IL 60532 USA</i> |
| Remove redundant address components | The input address string should not have repeated address components, such as two different organization names or repetitive name of an organization in one string.
Example: <i>Widget 2200 Western CT Widget Lisle IL 60532 USA</i> |
| Do not have merged components in address strings | Merged address components result in incorrect prediction.
Example:
<ul style="list-style-type: none"> • Incorrect: <i>Widget-Ltd-2200 Western CT Lisle IL 60532 USA</i> • Correct: <i>Widget Ltd 2200 Western CT Lisle IL 60532 USA</i> |
| Avoid addressee name in the string | Addressee name in the string results in incorrect prediction for the United States addresses.
Example:
<ul style="list-style-type: none"> • Incorrect: <i>Mr John Doe Widget 2200 Western CT Lisle IL 60532 USA</i> • Correct: <i>Widget 2200 Western CT Lisle IL 60532 USA</i> |
| Do not have bracketed "()" address component | Including any of your address components inside brackets "()" will leave it unparsed.
Example: <i>Widget 2200 Western CT (Lisle) IL 60532 USA</i> |

Limitations for United States Addresses

These are the limitations of the address parser for United States addresses:

- PO Box addresses are not supported.
- In Care of (C/O) addresses are not supported.
- If AddressNumber is missing, StreetNumber may be returned as AddressNumber (only in cases of numeric digits without superscripts).
- Direction may be returned in StateProvince for a few defined addresses (especially in cases where Direction is comprised of two letters).

Accessing Global Address Parser

When you install and deploy Spectrum Global Addressing Management, you will be able to use the Global Address Parser in these two ways:

- As a stage from the **Enterprise Designer**
- As a service from the **Management Console**

Using the **Address Parser** screen, you can perform these tasks:

1. **Set the parsing options:** Specify the country to which the addresses belong and the minimum confidence level required for parsing.
2. **Parse addresses:** Feed the address strings and get the parsed output.

Note: For details on how to perform these tasks, see the sections [Using Global Address Parser As a Stage](#) on page 540 and [Using Global Address Parser As a Service](#) on page 541.

Using Global Address Parser As a Stage

You can use Global Address Parser as an address parsing stage in your job. In this case, you can only perform batch address parsing.

To parse a batch of input address strings, append the Global Address Parser stage with an input and an output stage.

Note: For details on creating a job using any stage in the Enterprise Designer, see *My First Dataflow (Job)* in the *Dataflow Designer Guide*.

This table displays the Global Address Parser options.

Table 65: Global Address Parser Options

Field Name	Description
Override system default option with the following values	Select this check-box to change the default options.
Default Options	

Field Name	Description
Country	<p>Select the country to which the addresses to be parsed belong. The options are:</p> <ul style="list-style-type: none"> • Australia • Canada • France • Germany • Spain • United Kingdom • United States
Minimum confidence score	<p>On a scale of 0 to 100, assign the minimum confidence the parser should have on the result to display it.</p> <p>Note: Parsing results having a confidence score lesser than specified here are not displayed as output.</p>

Note: For details on the output fields, see [Parsed Address Output](#) on page 543.

Using Global Address Parser As a Service

To split your address strings into the appropriate components, you must perform these steps:

- Set the parsing options.
- Feed the input addresses to be parsed to the Global Address Parser.

To access the **Global Address Parser** screen and perform address parsing, follow these steps:

1. In a web browser, go to:

```
http://server:port/managementconsole
```

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


2. Log in with your credentials.
3. Under the **Services** tab, select **Global Addressing**.
4. From the list of services on the left side of the pane, select **Global Address Parser**. The **Global Address Parser** screen is displayed with the **Default Options** tab selected.
5. Select the **Country** to which the addresses you are parsing belong. You can parse addresses in these countries:
 - Australia
 - Canada
 - France

- Germany
 - Spain
 - United Kingdom
 - United States
6. Assign the **Minimum confidence score** you want for this parsing. The parser will not display results that fall below the confidence score you specify here.
 7. Click **Save**.
The options specified are saved for the next operation: Entering the addresses to be parsed.
 8. Click the **Preview** tab.
 9. Click one of these icons to feed the address that needs to be parsed.

- To add records one at a time to the parser, perform these tasks:


- a. Click the **Add record** button .
- b. In the **Address** field of the **Input Record <sequence of the address record>** section, enter the address string to be parsed.

Note: Repeat steps 'a' and 'b' to add multiple address strings. You can add up to 100 address strings.

- To import multiple addresses from a CSV file, click the **Import records** button . In the **Import Data** pop-up window that is displayed, enter these values:
 - a. In the **File name** field, select the file that has the address records.
 - b. Select the **Field separator** used in the address file.
 - c. Specify the **Maximum number of records to import**.
 - d. Click **OK**.

The entered or selected address records are displayed as input records below the **Run Preview** button.

Note: The input address string should have more than one token. For example, an input address string with `London` as the only value will not be recognized by the parser. An address need to have at least one more token (or component) to the string, such as city name, place name, or post code.

10. To delete any of the address strings, hover the cursor on the corresponding **Input Record <sequence of the address record>** and click the **Delete this record** icon  that is displayed.

Note: To delete all records, click **Delete all records**  icon placed below **Input Records**.

11. To view parsed output, click the **Run Preview** button.
The parsed address components (Output Record) are displayed adjacent to the corresponding input records. For details on the output fields, see [Parsed Address Output](#) on page 543.

Parsed Address Output

The parsed output displays a list of all the address components along with the corresponding values in the input address strings. The components for all countries are not identical. The table below gives a description of all the address components, the values the components can take, and if those are applicable for Australia (AUS), Canada (CAN), France (FRA), German (DEU), Spain (ESP), United Kingdom (GBR), and United States (USA) addresses.

Note: The values of some address components can be interchanged in the output:

- For Canada addresses: City and StateProvince can be interchanged.
- For German addresses: Suburb, City, County and StateProvince can be interchanged.
- For United Kingdom addresses: City, Suburb and Neighbourhood can be interchanged.

Table 66: Address Components, Definition, and Validity

Address Components	Valid for AUS	Valid for CAN	Valid for DEU	Valid for ESP	Valid for FRA	Valid for GBR	Valid for USA	Accepted and Parsed Values
OrganizationName	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Name of organization, hospital, institute, school, and bank.
Floor	-	-	-	-	-	Yes	-	Apartment number, sub-building information, floor, suite, and flat number.
PlaceName	-	-	-	-	-	Yes	-	Landmark, building, building name, cluster name, society name, residential and commercial complex, and special economic zone.
AddressNumber	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Building number, address number on streets.
Street	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Name of thoroughfare.
Neighbourhood	-	-	-	-	-	Yes	-	Small subdivision of a locality, city, or town.
City	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Name of village, city, district, or suburb, per the geographical division of the country. These elements together constitute location information.
Suburb	-	-	Yes	-	-	Yes	-	
County	-	-	Yes	-	-	Yes	-	

Address Components	Valid for AUS	Valid for CAN	Valid for DEU	Valid for ESP	Valid for FRA	Valid for GBR	Valid for USA	Accepted and Parsed Values
PostCode	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Series of letters and/or digits assigned to geographical areas primarily for sorting mails. Post Code sometimes also includes spaces or punctuation.
POBox	-	-	Yes	-	-	Yes	-	A lockable box having a unique address. It is located on the premises of a post office station.
StateProvince	Yes	Yes	Yes	Yes	Yes	-	Yes	Largest geographical entity with respect to a country.
Country	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Name of the country.

Note: All the components listed in the table may not be displayed for all the input addresses. For a component to display, the input string should have a value for it.

US Database Lookup

US Database Lookup provides the ability to search the US Database directly for address information. US Database Lookup is part of Spectrum Global Addressing Management.

Supported Countries

US Database Lookup provides search capability for the US database only.

Using US Database Lookup

After installing and deploying Spectrum Global Addressing Management, use the US Database Lookup to perform a:

- Last line lookup
- Street name lookup
- House number lookup
- ZIP Code lookup for a city or a city/state combination

To use US Database Lookup from Management Console:

1. Open the Management Console.
2. Under the Services tab, select **Global Addressing**.
3. From the list of services on the left side of the pane, select **US Database Lookup**.

4. On the **Database Resources** tab, specify the Global Address Validation database resource to use.
5. Click **Save** to save your database selection.
6. Click the **Default Options** tab.
7. On the **Default Options** tab, specify the maximum number of candidates you want returned. The default is 100.
8. If you make changes to the global default options, click **Save** to save those changes.
9. Click the **Preview** tab.
10. Follow the steps for the type of lookup you want to perform.
11. Click **Run Preview**.
12. In Preview **Output Records** on the right side of the pane, note that the results of the search have been placed in the appropriate output field. For information on the output fields.

Using Last Line Lookup for City, State, and ZIP Code

You can use Last Line Lookup to:

- Find all cities and ZIP Codes for a full or partial city and state
- Find all cities and ZIP Codes for a full or partial ZIP Code
- Find all cities for a ZIP Code and all ZIP Codes for a city

Using City and State for Last Line Lookup

To view all cities and ZIP Codes for a partial or full city and state:

1. In the LastLine field, enter a full or partial city and state.
2. Click **Run Preview**.
3. In Preview **Output Records** on the right side of the pane, note that the results of the search, based on the scope of the input provided, have been placed in the appropriate output fields.

Example: You enter "Whe" and "IL" in the LastLine field. The output records include all cities and ZIP codes for "Wheaton IL" and "Wheeling IL".

Using ZIP Code for Last Line Lookup

To view all cities and ZIP Codes for a full or partial ZIP Code:

1. In the LastLine field, enter a full or partial ZIP Code.
2. Click **Run Preview**.
3. In Preview **Output Records** on the right side of the pane, note that the results of the search, based on the scope of the input provided, have been placed in the appropriate output fields.

Example: You enter "6018" in the LastLine field. The output records include all ZIP Codes that start with the characters "6018" beginning with "60180" through "60189" and all the cities that correspond to the ZIP codes in that range.

Using City/State and ZIP Code for Last Line Lookup

To view all cities for a ZIP Code and all ZIP Codes for a City:

1. In the LastLine field, enter a full or partial City/State and ZIP Code.
2. Click **Run Preview**.
3. In Preview **Output Records** on the right side of the pane, note that the results of the search, based on the scope of the input provided, have been placed in the appropriate output fields.

Example: You enter "Wheaton IL 60187" in the LastLine field. The output records include "Wheaton IL 60187" and "Wheaton IL 60189". The output returns all cities for "60187" and all ZIP Codes for "Wheaton IL".

Using Last Line Lookup for Street Name

You can use Last Line Lookup to:

- Find all street names for a city/state
- Find all street names for a ZIP Code

Note: You must use a complete city/state and ZIP Code for a Street Name lookup.

Using City and State for Street Name Lookup

To view all street names for a city/state:

1. In the LastLine field, enter a full city and state.
2. In the StreetName field, enter "*" (asterisk without quotation marks).
3. Click **Run Preview**.
4. In Preview **Output Records** on the right side of the pane, note that the results of the search, based on the scope of the input provided, have been placed in the appropriate output fields.

Example: You enter "Wheaton" and "IL" in the LastLine field. You also enter "*" in the StreetName field. The output records include all street names found for "Wheaton IL".

Example: You enter "Wheaton" and "IL" in the LastLine field. You also enter "a" in the StreetName field. The output records include all street names that begin with "a" that are found for "Wheaton IL". You can also enter "and" to see all street names that begin with "and" that are found for "Wheaton IL".

Using ZIP Code for Street Name Lookup

To view all street names for a city/state:

1. In the LastLine field, enter a ZIP Code.
2. In the StreetName field, enter "*" (asterisk without quotation marks).

3. Click **Run Preview**.
4. In Preview **Output Records** on the right side of the pane, note that the results of the search, based on the scope of the input provided, have been placed in the appropriate output fields.

Example: You enter "60187" in the LastLine field. You also enter "*" in the StreetName field. The output records include all street names found for the ZIP Code "60187".

Example: You enter "60187" in the LastLine field. You also enter "a" in the StreetName field. The output records include all street names that begin with "a" that are found for the ZIP Code "60187". You can also enter "as" to see all street names that begin with "as" that are found for "60187".

Using Last Line Lookup for House Number

You can use Last Line Lookup to:

- Find all house numbers for a street name in a city/state
- Find all house numbers for a street name in a ZIP Code

Note: You must use a complete city/state and ZIP Code for the House Number lookup.

Using City and State for House Number Lookup

To view all house numbers for a street name in a city/state:

1. In the LastLine field, enter a full city and state.
2. In the StreetName field, enter a full street name.
3. In the HouseNumber field, enter "*" (asterisk without quotation marks).
4. Click **Run Preview**.
5. In Preview **Output Records** on the right side of the pane, note that the results of the search, based on the scope of the input provided, have been placed in the appropriate output fields.

Example: You enter "Wheaton" and "IL" in the LastLine field. You also enter "Lincoln" in the StreetName field and "*" in the HouseNumber field. The output records include all house numbers for Lincoln Ave in Wheaton IL.

Using ZIP Code for House Number Lookup

To view all house numbers for a street name in a city/state:

1. In the LastLine field, enter a full ZIP Code.
2. In the StreetName field, enter a full street name.
3. In the HouseNumber field, enter "*" (asterisk without quotation marks).
4. Click **Run Preview**.

5. In Preview **Output Records** on the right side of the pane, note that the results of the search, based on the scope of the input provided, have been placed in the appropriate output fields.

Example: You enter "60187" in the LastLine field. You also enter "Lincoln" in the StreetName field and "*" in the HouseNumber field. The output records include all house numbers for Lincoln Ave for the ZIP Code 60187.

Information Extraction stages

Read from Documents

Read from Documents is a source stage that reads unstructured input data from various file formats and extracts the contents. Possible sources include legal documents, customer feedback, product reviews, news articles, blogs, social networks, and so on. Read from Documents also extracts metadata fields such as author and creation date. Once the data has been extracted it can be used for various types of processing, including entity extraction and string manipulation among others. The data can also be used to build search indexes for unstructured text searches.

Note: Each document is considered one record for this stage.

Input

The input for Read from Documents is a single file or folder. This stage supports the following file types:

- Text
- PDF
- Microsoft Outlook
- Microsoft Word
- HTML

Read from Documents performs three types of extractions:

- Document—Use the entire document
- Page—Use a specific page of a document
- Selective—Use a selected part of a document
- Bookmarks—Use bookmarks from a PDF document

Read from Documents is part of Spectrum Entity Extraction.

Options

File Properties Tab

The following table lists the options that control the type of information returned by Read from Documents.

Table 67: Read from Documents Options

Option	Description								
Server name	Specifies the name of the Spectrum Technology Platform server being used.								
File/folder name	The path and name of the source document or folder. If you want to point to a folder, use an asterisk as a wildcard character ("*") to select all files in the folder. If you want to point to multiple files of the same type within a folder, use the wildcard character plus the file extension ("*.pdf").								
File type	The source document's file type, which will automatically be selected once you select a source: <ul style="list-style-type: none"> • Text • PDF • Microsoft Outlook • Microsoft Word • HTML 								
Extraction type	<table border="0"> <tr> <td>Documentation</td> <td>Use the entire document.</td> </tr> <tr> <td>Page</td> <td>Use a specific page of a document.</td> </tr> <tr> <td>Selection</td> <td>Use a selected portion of a document.</td> </tr> <tr> <td>Bookmarks</td> <td>Use bookmarks from a PDF document.</td> </tr> </table>	Documentation	Use the entire document.	Page	Use a specific page of a document.	Selection	Use a selected portion of a document.	Bookmarks	Use bookmarks from a PDF document.
Documentation	Use the entire document.								
Page	Use a specific page of a document.								
Selection	Use a selected portion of a document.								
Bookmarks	Use bookmarks from a PDF document.								
Page selection	Only with Page extraction type. Select all pages or a range of pages.								
Selected extraction	Only with Selection extraction type. Specifies the type of search.								

Option	Description
Specify text	Only with Selection extraction type. Specifies the text to look for.
Exclude start text	Only with Selection extraction type and Start text option. Omits entered string from the returned data.
Specify end text	Only with Selection extraction type. Specifies the end text to look for.
Exclude end text	Only with Selection extraction type. Omits entered string from the end of the returned data.
Selection return	Only with Selection extraction type. Specifies how many paragraphs to return for each result. For example, if you choose "2" here, the returned data for each result will include the paragraph the result is in plus the subsequent paragraph, totaling two paragraphs. Default is 1. Not valid when end text is specified.

Fields Tab

Click **Regenerate** to define input fields.

Table 68: Output Data Options

Option	Description
Attribute Name	Shows the attribute that is most like the input field. For instance, if one of your fields contains date information and you call it "Date," you will see the "Date" attribute assigned to that field. This column is not editable.
Name	The name of the field. This column is editable.
Type	The field's data type.
Include	Specifies which fields to be included in a search index.

Output

The Read from Documents stage has two outgoing ports. One port captures the data that was read by the stage and returned based on the criteria entered. It can include plain text or metadata (such as author, language, date created, and so on). This port can be connected to any stage that reads incoming data, such as Write to File or Write to XML, as well as primary stages such as Validate Address or Write to Search Index. It can also be connected to the Information Extractor stage if you want to return information about certain entity types that are in the document. When you select the Document extraction type the output will contain flat data; when you select the Page or Selection extraction type the output will contain hierarchical data.

The other port collects any records that the dataflow did not process correctly. This is called the Error Port, and records that pass through this port into the sink are considered malformed. Capturing malformed records can help you identify the problem with those records. When you attach a sink to the Error Port, the resulting output file will contain all the fields from the malformed records. It will also contain a Reason field that specifies why the record failed.

Table 69: Unstructured Reader Output

Field Name	Description / Valid Values				
Author	Typically contains the name of the person who created or updated the document. This information is part of the document's metadata.				
Bookmark	Contains all the bookmarks from the PDF input file. For Bookmarks extraction types only.				
BookmarkNo	Contains all the bookmarks from the PDF input file. For Bookmarks extraction types only.				
ContentLength	Indicates the length of the document. This value varies depending on the extraction type selected: <table border="0" data-bbox="548 1549 1433 1627"> <tr> <td>Document</td> <td>The number of pages in the document.</td> </tr> <tr> <td>Page</td> <td>"1", to represent the single page of content.</td> </tr> </table>	Document	The number of pages in the document.	Page	"1", to represent the single page of content.
Document	The number of pages in the document.				
Page	"1", to represent the single page of content.				
Contents	Varies based on extraction type. For example, Document extraction types will output the entire document as flat data. Page, Selection, and Bookmarks extraction types will output hierarchical data.				

Field Name	Description / Valid Values
ContentType	Indicates the type of document that was read, such as PDF, .txt, and so on.
Creator	Typically contains the name of the person who created the document. This information is part of the document's metadata.
Date	Indicates the date the document was created or last updated.
Keywords	Contains any keywords that were provided in the document's metadata.
Language	Indicates the language in which the document was written.
NPages	Indicates the number of pages in the document.
PageContents	Contains the contents of the selected page(s). For Page extraction types only.
PageNo	Contains the page number for the bookmark. For Page extraction types only.
Parent	Contains the path of the bookmark, similar to XPath of an XML file. For Bookmarks extraction types only.
ResourceName	Indicates the file name of the document.
SectionContents	Contains the contents of the selected section. For Selection extraction types only.
SectionNo	Indicates the number of that section within the document. For Selection extraction types only.
Subject	Contains the subject of the document that was provided in the document's metadata.
Title	Contains the title of the document that was provided in the document's metadata.

Entity Extractor

Entity Extractor extracts entities such as names and addresses from strings of unstructured data (also known as plain text).

It is possible that not all entities for any selected type will be returned because accuracy varies depending on the type of input. Because Entity Extractor uses natural-language processing, a string containing a grammatically correct sentence from a news article or blog would likely have a more accurate return of names than a simple list of names and dates.

Input

Entity Extractor takes unstructured strings of data as the input. It can also use the **Read from Documents** stage as an input if you want to extract entities from an unstructured document. The **Read from Documents** stage reads the document and returns text based on the user-defined settings. The **Entity Extractor** extracts the required information from this text based on the selected entities.

Table 70: Input Format

Field Name	Description
PlainText	The unstructured string of data from which you want to extract information.

Options

Entity Extractor options enable you to select entities based on which you want to extract information from the input string. By default, you can extract information using *Person* and *Address* as the entity types. However, you can use the **Quick Add** function and select any or all of the 15 pre-configured entities.

Option Name	Description
Override system default options with the following values	<p>Select the check box to override the default entity types <i>Address</i> and <i>Person</i>.</p> <p>When you select the check box, the Quick Add button gets enabled. Click this button and select the entities you need for extracting the text.</p> <p>The selected entities get added to the Entity Type list.</p>

Option Name	Description
Entity Type	<p>Specifies the type of data you want to extract from the unstructured string.</p> <p>Address</p> <p>CreditCard</p> <p>Date</p> <p>Email</p> <p>HashTag</p> <p>ISBN</p> <p>Location</p> <p>Mention</p> <p>Organization</p> <p>Person</p> <p>Phone</p> <p>ProperNouns</p> <p>SSN</p> <p>WebAddress</p> <p>ZipCode</p>
Output entities count	<p>Specifies whether to return a count of the number of times a particular entity occurred in the output.</p> <p>true Return a count of the entities found in the unstructured string.</p> <p>false Do not return a count of the entities found in the unstructured string.</p>

Output

The output from **Entity Extractor** is a list of the matching entities found in the input string. For example, if you selected an entity type as "Person," the output would contain a list of the person names found in the input string. Similarly, if you selected the **Entity Type** as "Date," the output will be a list of the dates found in the input string.

Each entity, whether a name, address, or date, is returned only once even if the entity appears multiple times in the input string.

To see the number of times the entity appeared in the input string you can select the **Output entity count** option in the **Entity Extractor Options** window.

Field Name	Description
Text	The text extracted from the string.
Type	The entity type of the extracted text. One of the following: Address CreditCard Date Email HashTag ISBN Location Mention Organization Person Phone ProperNouns SSN WebAddress ZipCode
Count	If the option to return a count is enabled, this field contains the number of times a particular entity appeared in the input. For example, if you chose to return <code>Name</code> entities and the input text contained five instances of the name <code>John</code> , the name <code>John</code> will be included in the output just once, with <code>Name</code> as the entity type, and "5" as the output count.

Relationship Extractor

The **Relationship Extractor** stage allows you to identify the relationship types between the identified entities in the source content.

The **Relationship Extractor** stage identifies:

1. Entity1
2. Entity1 Type
3. Relation Type
4. Entity2
5. Entity2 Type

Important: The stage tries to achieve the maximum possible accuracy while identifying the relationship types between any two entities in the input text. However, relationships other than the accurate relationship between the two entities can also be identified while parsing complicated sentences in the input text.

Input

The **Relationship Extractor** stage takes natural language strings of data as the input, and identifies the entities and the relationship types existing between each pair of entities.

Use the **Read from Documents** stage as a source stage if the input text is from an unstructured document. The **Read from Documents** stage reads the document and returns text based on the user-defined settings.

The **Relationship Extractor** stage then identifies all the entities and the relationship type existing between each pair of entities.

Table 71: Input Format

Field Name	Description
PlainText	The unstructured string of data from which you want to identify the relationship types existing between each pair of entities.

Options

The **Relationship Extractor** stage options enable you to specify which relationship types you wish to identify in the input text.

By default, the relationship types identified are:

1. *AffiliatedWith*
2. *LivesIn*
3. *OrgBasedIn*
4. *LocatedIn*

Option Name	Description
Override system default options with the following values	<p>Select the check box to override the default relationship types identified and specify which relationship types you wish to identify and extract from the input text.</p> <p>On selecting the check box, the Quick Add button is enabled. Click Quick Add to select the relationship types you wish to identify in the text.</p> <p>The selected entities get added to the Relationship Type list.</p>

Output

The output from **Relationship Extractor** is a list of the sets of relationships identified between pairs of entities found in the input string.

For example, if in the stage options, you have selected the *LivesIn* and *OrgBasedIn* relationship types to be extracted, the output contains a list of the all the sets of *Person LivesIn Location* and *Organization OrgBasedIn Location* identified in the input text.

Each entity pair with its binding relationship type is listed only once.

For each extracted set of entities and their relationship, the information extracted is:

Field Name	Description
Entity1	The first entity of a pair of entities extracted from the input text.
Entity1 Type	<p>The entity type of the first entity of the pair of entities extracted from the input text.</p> <p>The entity type is any one of these:</p> <ul style="list-style-type: none"> • Person • Organisation • Location
Type	<p>The relationship type identified between Entity1 and Entity2.</p> <p>For more information about relationship types, see Relationship Types.</p> <p>Note: Only the relationship types selected for extraction in the stage options are identified and listed.</p>
Entity2	The second entity of a pair of entities extracted from the input text.

Field Name	Description
Entity2 Type	<p>The entity type of the second entity of the pair of entities extracted from the input text.</p> <p>The entity type is any one of these:</p> <ul style="list-style-type: none"> • Person • Organisation • Location

Text Categorizer

This stage helps you assign custom categories to unstructured content or plain text (such as email, news articles, and comments) based on the extent of matching content it has. The stage lists the defined categories, from which you can select the one you need for your categorization. However, you need to create these categories by training a categorizer model with your data. For details, see [Text Categorizer](#).

Input

The stage takes unstructured strings of data as input . It can also use the **Read from Documents** stage as an input if you want to categorize text from an unstructured document. The **Read from Documents** stage reads the document and returns text based on the user-defined settings. This is read by the **Text Categorizer** stage to give you the desired output.

Table 72: Input Format

Field Name	Description
PlainText	The unstructured string of data from which you want to extract information.

Options

The **Text Categorizer Options** gives you the choice of selecting parameters based on which you want to classify your input string of data. You can select the model for categorization and the number of matching levels to which you want the output. For example, only the closest match or closest plus the second close match.

Option Name	Description
Override system default options with the following values	To override the default option and select the categorizer from the Categorizer name drop down.
Categorizer name	Specifies the model to be used for text categorization. It lists all the models you trained in the text categorization phase. Note: For more information, see Training the Model .
Category count	The count of matching levels of category that you want in the output. For example, select 1 to display just the closest match and 2 to display the closest plus the second close match. Note: The maximum value corresponds to the number of different classes specified while training the model.

Output

The output lists the categories into which the content of the input string are classified and the rank of that category. The rank signifies how close the input content matches the category. For example, 1 means it is the closest match to the category and 2 means closest plus the next close match.

Field Name	Description
Category	The predicted category for each record in the input file.
Rank	The rank of categories from the highest score to the lowest score.

Machine Learning Stages

Binning

Introduction

The Binning stage performs what is known as unsupervised binning, which divides a continuous variable into groups (bins) without taking into account objective information. The data captured includes ranges, quantities, and percentage of values within each range.

Advantages to performing binning include the following:

- It allows records with missing data to be included in the model.
- It controls or mitigates the impact of outliers over the model.
- It solves the issue of having different scales among the characteristics, making the weights of the coefficients in the final model comparable.

In Spectrum Technology Platform unsupervised binning, you can use equal-width bins, where the data is divided into bins of equal size, or equal-frequency bins, where the data is divided into groups containing approximately the same number of records. In the Binning stage, equal-width bins are referred to as Equal Range bins and equal-frequency bins are referred to as Equal Population bins.

You can perform more binning functions using the Machine Learning Model Management [Binning Management](#) tool.

You can also view a list of binning and delete binning using command line instructions. See "Binning" in the "Administration Utility" section of the *Administration Guide*.

Defining Binning Properties

1. Under **Primary Stages** > **Deployed Stages** > **Machine Learning**, click the **Binning** stage and drag it onto the canvas, placing it where you want on the dataflow and connecting it to other stages.

Note: The input stage must be the data source that contains both the objective and input variable fields for your model. An output stage is not required unless you select the **Score input data** option on the **Basic Options** tab. You may also connect an output stage if you wish to capture your output independent of the Machine Learning Model Management tool.

2. Double-click the Binning stage to show the **Binning Options** dialog box.
3. Enter a **Binning name** if you do not want to use the default name.
4. Check the **Overwrite** box to overwrite the existing model with new data.
5. Enter a **Description** of the model.

6. Click **Include** for each field whose data you want included in binning. Note that only numeric fields will appear in this list.
7. Click **OK** to save your settings.

Configuring Basic Options

1. Select whether you want to perform an equal-range or equal-population **Binning style**.
2. Select in **Null value bin** how you want to handle empty bin fields, which represent unknown values due to missing data.
 - Select **Highest** to assign null values to the highest bin.
 - Select **Lowest** to assign null values to the lowest bin.

The lowest bin is always bin 1.

3. Click **Target internal bins** and enter the number of bins you want to fill between the end bins. If you are performing equal-range binning, you may select this type of processing or **Bin width**, but not both. If you are performing equal-population binning, you may only perform internal-bin processing.
4. If you are performing equal-range binning and want to select this type of processing rather than internal-bin processing, click **Bin width** and enter the number of units you want in each bin.
5. Click **Include** for each field whose data you want included in binning.

Note: Only numeric fields will appear in this list.

6. Click **OK** to save your settings.

Binning Output

The Binning stage has two output ports. The first port will output all input fields plus a binned field for each selected input field. For example, if the input contains Name, Age, and Income fields and you perform binning on Age and Income, the output from the first port will contain the following fields:

- Name
- Age
- Binned_Age
- Income
- Binned_Income

The second port outputs four types of information for each selected input field. For example, if you perform binning on Age, the output from the second port will contain the following fields:

- Age_Bins
- Age_BinValue
- Age_Count
- Age_Percentage

K-Means Clustering

Introduction

K-Means Clustering creates models based on analytical clustering, which segments a set of records into clusters of similar records based on data values.

To create your model, you must first complete the **Model Properties** tab. The **Basic Options** and **Advanced Options** tabs provide sufficient default settings to complete a job, but you can alter those settings to meet your needs. You then run your job and a limited version of the resulting model output details appears on the **Model Output** tab. The model is stored on the Spectrum Technology Platform server and the complete output is available in the Machine Learning Model Management tool.

Defining Model Properties

1. Under **Primary Stages > Deployed Stages > Machine Learning**, click the **K-Means Clustering** stage and drag it onto the canvas, placing it where you want on the dataflow and connecting it to other stages.

Note: The input stage must be the data source that contains input variable fields for your model. An output stage is not required unless you select the **Score input data** option on the **Basic Options** tab. You may also connect an output stage if you wish to capture your output independent of the Machine Learning Model Management tool.

2. Double-click the K-Means stage to show the **K-Means Clustering Options** dialog box.
3. Enter a **Model name** if you do not want to use the default name.
4. Optional: Check the **Overwrite** box to overwrite the existing model with new data.
5. Enter the **Number of clusters** you want in your model if you do not want the default number (5).
6. Optional: Enter a **Description** of the model.
7. Click **Include** for each field whose data you want added to the model.
8. Use the **Model Data Type** drop-down to specify whether the input field is to be used as a numeric, categorical, or datetime field.
9. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Basic Options

1. Leave **Standardize input fields** checked to standardize the numeric columns to have zero mean and unit variance.

If you do not use standardization, the results may include components dominated by variables appearing to have larger variances relative to other attributes as a matter of scale rather than true contribution.

2. Check **Estimate number of clusters** to have the K-Means algorithm attempt to determine the number of clusters that your model will contain. Even though you designate the number of desired

clusters on the Model Properties tab, the routine may discover in its processing that a different number of clusters is more appropriate given the data.

- Specify a value between 1 and 100 as the **Percentage for training data** when the input data is randomly split into training and test data samples.
- Enter the value of 100 minus the amount you entered in step 3 on page 563 as the **Percentage for test data**.
- Enter a number as the **Seed for sampling** to ensure that when the data is split into test and train data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.
- Click **OK** to save the model and configuration or continue to the next tab.

Configuring Advanced Options

- Leave **Ignore constant fields** checked to skip fields that have the same value for each record.
- Leave **Seed for algorithm** checked and enter a seed number to ensure that when the data is split into test and training data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.
- Select the correct initialization mode in the **Init** dropdown.

Initialization mode	Description
Furthest	Initializes the first centroid randomly, but then initializes the second centroid to be the data point farthest away from it. Initializes the centroids to be well spread-out from each other.
Plus-Plus	Initializes the cluster centers before proceeding with the standard k -means optimization iterations. With the k -means++ initialization, the algorithm is guaranteed to find a solution that is $O(\log k)$ competitive to the optimal k -means solution.
Random	Chooses K clusters from the set of N observations at random so that each observation has an equal chance of being chosen. This is the default initialization mode.

- Leave **Seed for N fold** checked and enter a seed number to ensure that when the data is split into test and train data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.
- Check **N fold** and enter the number of folds if you are performing cross-validation.
- Check **Fold assignment** and select from the drop-down list if you are performing cross-validation.

Fold assignment	Description
Auto	Allows the algorithm to automatically choose an option; currently it uses Random. This is the default.
Modulo	Evenly splits the dataset into the folds and does not depend on the seed.

Note: This field is applicable only if you entered a value in **N fold**.

7. Check **Maximum iterations** and enter the number of training iterations that should take place.
8. Click **OK** to save the model and configuration or continue to the next tab.

Model Output

This tab shows the metrics you are using to assess the fitted model. These fields cannot be edited. The **Training** column will always contain data. If you selected a train/test split on the **Basic Options** tab, the **Test** column will also be filled, unless you have selected an N Fold validation on the **Advanced Options** tab, in which case the **N Fold** column will be filled. Click the **Output** button to regenerate the output, and click **Model details** to view the entire output in the Machine Learning Model Management tool.


Output Port

The K-Means Clustering stage contains one optional output port: the Model Metrics Port. This port's functionality is determined by your selections and input when completing the stage's basic and advanced options. For example, if you choose to conduct N Fold validation by checking the **N Fold** field on the **Advanced Options** tab, the **N Fold** column in the output metrics will be populated with data. Alternatively, if you choose not to conduct N Fold validation, the **N Fold** column will be blank.

Model Metrics Port

Perform this procedure to use the Model Metrics Port.

The **Model Metrics Port** lets you output the model assessment metrics to a data file. This will help you compare many models generated from within and outside of Spectrum Technology Platform and perform other data processing tasks on the metrics.

1. Open a dataflow that uses the K-Means Clustering stage.
2. Attach a Write to File stage or another data output stage to the second output port.
3. Run the job.
4. Alternative to step 3 on page 564: Add an inspection point to the channel that connects the K-Means Clustering stage to the sink stage you added in step 2 on page 564 by right-clicking the channel and selecting "Add inspection point." Then click the Inspect Current Flow button  on the Enterprise Designer toolbar.

Inspection will run and you should see results similar to those shown below.

Creation Time	Flow Name	Metrics	Model Name	Model Type	N Fold	Test	Training
10/11/2018 2:37:00 AM	LinearRegressionTest1...	MSE	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10
10/11/2018 2:37:00 AM	LinearRegressionTest1...	RMSE	LinearRegressionTest1...	Linear Regression	0.0707247064967455	0.0567287296394643	1.14687396359675E-05
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Number of observations	LinearRegressionTest1...	Linear Regression	33	17	33
10/11/2018 2:37:00 AM	LinearRegressionTest1...	R2	LinearRegressionTest1...	Linear Regression	-0.366538763835926	0.0268719304981138	0.999999964065547
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Mean residual deviance	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10

Linear Regression

Introduction

Linear Regression enables you to perform machine learning by creating models from datasets that use continuous objectives with input variables.

To create your model, you must first complete the **Model Properties** tab. The **Basic Options** and **Advanced Options** tabs provide sufficient default settings to complete a job, but you can change those settings to meet your needs. You then run your job and a limited version of the resulting model appears on the **Model Output** tab; the complete output is available in the Machine Learning Model Management tool.

Defining Model Properties

1. Under **Primary Stages > Deployed Stages > Machine Learning**, click the **Linear Regression** stage and drag it onto the canvas, placing it where you want on the dataflow and connecting it to other stages. Note that the input stage must be the data source that contains both the objective and input variable fields for your model; an output stage is not required unless you select the Score input data option on the Basic Options tab. You may also connect an output stage if you wish to capture your output independent of the Machine Learning Model Management tool.
2. Double-click the Linear Regression stage to show the **Linear Regression Options** dialog box.
3. Enter a **Model name** if you do not want to use the default name.
4. Check the **Overwrite** box to overwrite the existing model with new data.
5. Click the **Objective field** drop-down and select a numerical field.
6. Enter a **Description** of the model.
7. Click **Include** for each field whose data you want added to the model; be sure to include the field you selected as the Objective field.
8. Use the **Model Data Type** drop-down to specify whether each input field is to be used as a numeric, categorical, or datetime field.
9. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Basic Options

1. Leave **Standardize input fields** checked to standardize the numeric columns to have zero mean and unit variance.

If you do not use standardization, the results may include components dominated by variables appearing to have larger variances relative to other attributes as a matter of scale rather than true contribution.

2. Check **Score input data** to add a column for the model prediction (score) to the input data.
3. Select a **Link function** from the drop-down list. This specifies the link between random and systematic components. It says how the expected value of the response relates to the linear predictor of explanatory variables.

Link function	Description
Identity	Predicts nonsense "probabilities" less than zero or greater than one; sometimes used for binomial data to yield a linear probability model. $g(p) = p$
Inverse	Computes the inverse of link functions for real estimates. $g(\mu_i) = 1/\mu_i$
Log	Counts occurrences in a fixed amount of time and space. $g(\mu_i) = \log(\mu_i)$

4. Specify how to handle missing data by checking **Skip** or **Impute means**, which will add the mean value for any missing data.
5. Specify a value between 1 and 100 as the **Percentage for training data** when the input data is randomly split into training and test data samples.
6. Enter the value of 100 minus the amount you entered in step 5 on page 566 as the **Percentage for test data**.
7. Enter a number as the **Seed for sampling** to ensure that when the data is split into test and train data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.
8. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Advanced Options

1. Leave **Ignore constant fields** checked to skip fields that have the same value for each record.
2. Check **Compute p values** to calculate p values for the parameter estimates.
3. Check **Remove collinear column** to automatically remove collinear columns during model building.

This option must be checked if **Compute p values** is also checked.

This will result in a 0 coefficient in the returned model.

4. Leave **Include constant term (Intercept)** checked to include a constant term (intercept) in the model.

This field must be checked if **Remove collinear column** is also checked.

5. Select a **Solver** from the dropdown list.

Solver	Description
Auto	Solver will be determined based on input data and parameters.
CoordinateDescent	IRLSM with the covariance updates version of cyclical coordinate descent in the innermost loop.
CoordinateDescentNaive	IRLSM with the naive updates version of cyclical coordinate descent in the innermost loop.
IRLSM	Ideal for problems with a small number of predictors or for Lambda searches with L1 penalty.

Note: **CoordinateDescent** and **CoordinateDescentNaive** are currently experimental.

6. Leave **Seed for N fold** checked and enter a seed number to ensure that when the data is split into test and train data it will occur the same way each time you run the dataflow. Uncheck in this field to get a random split each time you run the flow.
7. Check **N fold** and enter the number of folds if you are performing cross-validation.
8. Click **Fold assignment** and select from the drop-down list if you are performing cross-validation. This field is applicable only if you entered a value in **N fold** and **Fold field** is not specified.

Option	Description
Auto	Allows the algorithm to automatically choose an option; currently it uses Random.
Modulo	Evenly splits the dataset into the folds and does not depend on the seed.
Random	Randomly splits the data into nfolds pieces; best for large datasets.

9. If you are performing cross-validation, check **Fold field** and select the field that contains the cross-validation fold index assignment from the drop-down list.

This field is applicable only if you did not enter a value in **N fold** and **Fold assignment**.

10. Check **Maximum iterations** and enter the number of training iterations that should take place.
11. Check **Objective epsilon** and enter the threshold for convergence; this must be a value between 0 and 1.
If the objective value is less than this threshold, the model will be converged.
12. Check **Beta epsilon** and enter the threshold for convergence; this must be a value between 0 and 1.
If the objective value is less than this threshold, the model will be converged. If the L1 normalization of the current beta change is below this threshold, consider using convergence.
13. Select the **Regularization type** you want to use.

Regularization type	Description
LASSO (Least Absolute Shrinkage and Selection Operator)	Selects a small subset of variables with a value of lambda high enough to be considered crucial. May not perform well when there are correlated predictor variables, as it will select one variable of the correlated group and remove all others. Also limited by high dimensionality; when a model contains more variables than records, LASSO is limited in how many variables it can select. Ridge Regression does not have this limitation. When the number of variables included in the model is large, or if the solution is known to be sparse, LASSO is recommended.
Ridge Regression	Retains all predictor variables and shrinks their coefficients proportionally. When correlated predictor variables exist, Ridge Regression reduces the coefficients of the entire group of correlated variables towards equaling one another. If you do not want correlated predictor variables removed from your model, use Ridge Regression.
Elastic Net	Combines LASSO and Ridge Regression by acting as a variable selector while also preserving the grouping effect for correlated variables (shrinking coefficients of correlated variables simultaneously). Elastic Net is not limited by high dimensionality and can evaluate all variables when a model contains more variables than records.

A common concern in predictive modeling is overfitting, when an analytical model corresponds too closely (or exactly) to a specific dataset and therefore may fail when applied to additional data or future observations. Regularization is one method used to mitigate overfitting.

14. Check **Value of alpha** and change the value if you do not want to use the default of .5.
The alpha parameter controls the distribution between the ℓ_1 and ℓ_2 penalties. Valid values range between 0 and 1; a value of 1.0 represents LASSO, and a value of 0.0 produces ridge regression. The table below illustrates how alpha and lambda affect regularization.

lambda value	alpha value	Result
lambda == 0	alpha = any value	No regularization. alpha is ignored.
lambda > 0	alpha == 0	Ridge Regression
lambda > 0	alpha == 1	LASSO
lambda > 0	0 < alpha < 1	Elastic Net Penalty

Note: The single equals sign is an assignment operator meaning "is," while the double equals sign is an equality operator meaning "equal to."

15. Check **Value of lambda** and specify a value if you do not want Linear Regression to use the default method of calculating the lambda value, which is a heuristic based on training data.
The lambda parameter controls the amount of regularization applied. For instance, if lambda is 0.0, no regularization is applied and the alpha parameter is ignored.
16. Check **Search for optimal value of lambda** to have Linear Regression compute models for full regularization path.
This starts at lambda max (the highest lambda value that makes sense—that is, the lowest value driving all coefficients to zero) and goes down to lambda min on the log scale, decreasing regularization strength at each step. The returned model will have coefficients corresponding to the optimal lambda value as decided during training.
17. Check **Stop early** to end processing when there is no more relative improvement on the training or validation set.
18. Check **Maximum lambdas to search** and enter the maximum number of lambdas to use during the process of lambda search.
19. Check **Maximum active predictors** and enter the maximum number of predictors to use during computation.
This value is used as a stopping criterion to prevent expensive model building with many predictors.
20. Click **OK** to save the model and configuration or continue to the next tab.

Model Output

This tab shows the metrics you are using to assess the fitted model. These fields cannot be edited. The Training column will always contain data. If you selected a train/test split on the Basic Options tab, the Test column will also be filled, unless you have selected an N Fold validation on the Advanced Options tab, in which case the N Fold column will be filled.

After you run your job, the resulting model is stored on the Spectrum Technology Platform server. Click the **Output** button to regenerate the output and click **Model details** to view the entire output in the Machine Learning Model Management tool.

Output Ports

The Linear Regression stage contains two optional output ports: the Model Score Port and the Model Metrics Port. The functionality of these ports is determined by your selections and input when completing the stage's basic and advanced options. For example, if you choose to conduct N Fold validation by checking the **N Fold** field on the Advanced Options tab, the N Fold column in the output metrics generated by the Model Metrics Port will be populated with data. Alternatively, if you choose not to conduct N Fold validation, the N Fold column will be blank. Likewise, The Model Score Port is activated when you check the **Score input data** field on the Basic Options tab.


Model Score Port

When you check the **Score input data** field on the Basic Options tab, this tells Linear Regression to calculate predicted values when creating the model, which in turn adds the **Predicted_Value** column for that score in the output data. You can attach any kind of sink to this port: a Write to File stage, a Write to Null stage, and so on.

Model Metrics Port

Follow steps in this procedure to use the Model Metrics Port.

The **Model Metrics Port** lets you output the model assessment metrics to a data file. This will help you compare many models generated from within and outside of Spectrum Technology Platform and perform other data processing tasks on the metrics.

1. Open a dataflow that uses the Linear Regression stage.
2. Attach a Write to File stage or another data output stage to the second output port.
3. Run the job.
4. Alternative to step 3 on page 570: Add an inspection point to the channel that connects the Linear Regression stage to the sink stage you added in step 2 by right-clicking the channel and selecting "Add inspection point." Then click the Inspect Current Flow button  on the Enterprise Designer toolbar. Inspection will run and you should see results similar to the ones shown below.

Creation Time	Flow Name	Metrics	Model Name	Model Type	N Fold	Test	Training
10/11/2018 2:37:00 AM	LinearRegressionTest1...	MSE	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10
10/11/2018 2:37:00 AM	LinearRegressionTest1...	RMSE	LinearRegressionTest1...	Linear Regression	0.0707247064967455	0.0567287296394643	1.14687396359675E-05
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Number of observations	LinearRegressionTest1...	Linear Regression	33	17	33
10/11/2018 2:37:00 AM	LinearRegressionTest1...	R2	LinearRegressionTest1...	Linear Regression	-0.366538763835926	0.0268719304981138	0.999999964065547
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Mean residual deviance	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10

Logistic Regression

Introduction

Logistic Regression enables you to perform machine learning by creating models from datasets that use binary objectives with input variables.

To create your model, you must first complete the **Model Properties** tab. The **Basic Options** and **Advanced Options** tabs provide sufficient default settings to complete a job, but you can change those settings to meet your needs. You then run your job and a limited version of the resulting model

appears on the **Model Output** tab. The complete output is available in the Machine Learning Model Management tool.

Defining Model Properties

1. Under **Primary Stages > Deployed Stages > Machine Learning**, click the **Logistic Regression** stage and drag it onto the canvas, placing it where you want on the dataflow and connecting it to other stages.

Note: The input stage must be the data source that contains both the objective and input variable fields for your model; an output stage is not required unless you select the **Score input data** option on the **Basic Options** tab. You may also connect an output stage if you wish to capture your output independent of the Machine Learning Model Management tool.

2. Double-click the Logistic Regression stage to show the **Logistic Regression Options** dialog box.
3. Enter a **Model name** if you do not want to use the default name.
4. Check the **Overwrite** box to overwrite the existing model with new data.
5. Click the **Objective field** drop-down and select "Categorical."
6. Enter a **Description** of the model.
7. Click **Include** for each field whose data you want added to the model.
Be sure to include the field you selected as the Objective field.
8. Use the **Model Data Type** drop-down to specify whether each input field is to be used as a numeric, categorical, or datetime field.
9. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Basic Options

1. Leave **Standardize input fields** checked to standardize the numeric columns to have zero mean and unit variance.
If you do not use standardization, the results may include components dominated by variables appearing to have larger variances relative to other attributes as a matter of scale rather than true contribution.
2. Check **Score input data** to add a column for the model prediction (score) to the input data.
3. Check **Prior** if the data has been sampled and the mean of response does not reflect reality; then enter the prior probability for $p(y=1)$ in the text field.
4. Specify how to handle missing data by checking **Skip** or **Impute means**, which will add the mean value for any missing data.
5. Specify a value between 1 and 100 as the **Percentage for training data** when the input data is randomly split into training and test data samples.
6. Enter the value of 100 minus the amount you entered in Step 5 as the **Percentage for test data**.

- Enter a number as the **Seed for sampling** to ensure that when the data is split into test and train data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.
- Click **OK** to save the model and configuration or continue to the next tab.

Configuring Advanced Options

- Leave **Ignore constant fields** checked to skip fields that have the same value for each record.
- Leave **Compute p values** checked to calculate p values for the parameter estimates.
- Leave **Remove collinear column** checked to automatically remove collinear columns during model building.
This option must be checked if **Compute p values** is also checked.
This will result in a 0 coefficient in the returned model.
- Leave **Include constant term (Intercept)** checked to include a constant term (intercept) in the model.
This field must be checked if **Remove collinear column** is also checked.
- Select a **Solver** from the dropdown list.

Solver	Description
Auto	Solver will be determined based on input data and parameters.
CoordinateDescent	IRLSM with the covariance updates version of cyclical coordinate descent in the innermost loop.
CoordinateDescentNaive	IRLSM with the naive updates version of cyclical coordinate descent in the innermost loop.
IRLSM	Ideal for problems with a small number of predictors or for Lambda searches with L1 penalty.
L_BFGS	Ideal for datasets with many columns.

Note: **CoordinateDescentNaive** and **CoordinateDescentNaive** are currently experimental.

- Leave **Seed for N fold** checked and enter a seed number to ensure that when the data is split into test and train data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.
- Check **N fold** and enter the number of folds if you are performing cross-validation.
- Check **Fold assignment** and select from the drop-down list if you are performing cross-validation.

Fold assignment	Description
Auto	Allows the algorithm to automatically choose an option; currently it uses Random.
Modulo	Evenly splits the dataset into the folds and does not depend on the seed.
Random	Randomly splits the data into nfolds pieces; best for large datasets.
Stratified	Stratifies the folds based on the response variable for classification problems. Evenly distributes observations from the different classes to all sets when splitting a dataset into train and test data. This can be useful if there are many classes and the dataset is relatively small.

This field is applicable only if you entered a value in **N fold** and **Fold field** is not specified.

- If you are performing cross-validation, check **Fold field** and select the field that contains the cross-validation fold index assignment from the drop-down list.

This field is applicable only if you did not enter a value in **N fold** and **Fold assignment**.

- Check **Maximum iterations** and enter the number of training iterations that should take place.
- Check **Objective epsilon** and enter the threshold for convergence; this must be a value between 0 and 1.
If the objective value is less than this threshold, the model will be converged.
- Check **Beta epsilon** and enter the threshold for convergence; this must be a value between 0 and 1.
If the objective value is less than this threshold, the model will be converged. If the L1 normalization of the current beta change is below this threshold, consider using convergence.
- Select the **Regularization type** you want to use.

Regularization type	Description
LASSO (Least Absolute Shrinkage and Selection Operator)	Selects a small subset of variables with a value of lambda high enough to be considered crucial. May not perform well when there are correlated predictor variables, as it will select one variable of the correlated group and remove all others. Also limited by high dimensionality; when a model contains more variables than records, LASSO is limited in how many variables it can select. Ridge Regression does not have this limitation. When the number of variables included in the model is large, or if the solution is known to be sparse, LASSO is recommended.
Ridge Regression	Retains all predictor variables and shrinks their coefficients proportionally. When correlated predictor variables exist, Ridge Regression reduces the

Regularization type	Description
	coefficients of the entire group of correlated variables towards equaling one another. If you do not want correlated predictor variables removed from your model, use Ridge Regression.
Elastic Net	Combines LASSO and Ridge Regression by acting as a variable selector while also preserving the grouping effect for correlated variables (shrinking coefficients of correlated variables simultaneously). Elastic Net is not limited by high dimensionality and can evaluate all variables when a model contains more variables than records.

A common concern in predictive modeling is overfitting, when an analytical model corresponds too closely (or exactly) to a specific dataset and therefore may fail when applied to additional data or future observations. Regularization is one method used to mitigate overfitting.

14. Check **Value of alpha** and change the value if you do not want to use the default of .5.

The alpha parameter controls the distribution between the ℓ_1 and ℓ_2 penalties. Valid values range between 0 and 1; a value of 1.0 represents LASSO, and a value of 0.0 produces ridge regression. The table below illustrates how alpha and lambda affect regularization.

lambda value	alpha value	Result
lambda == 0	alpha = any value	No regularization. alpha is ignored.
lambda > 0	alpha == 0	Ridge Regression
lambda > 0	alpha == 1	LASSO
lambda > 0	0 < alpha < 1	Elastic Net Penalty

Note: The single equals sign is an assignment operator meaning "is," while the double equals sign is an equality operator meaning "equal to."

15. Check **Value of lambda** and specify a value if you do not want Logistic Regression to use the default method of calculating the lambda value, which is a heuristic based on training data. The lambda parameter controls the amount of regularization applied. For example, if lambda is 0.0, no regularization is applied and the alpha parameter is ignored.
16. Check **Search for optimal value of lambda** to have Logistic Regression compute models for full regularization path. This starts at lambda max (the highest lambda value that makes sense—that is, the lowest value driving all coefficients to zero) and goes down to lambda min on the log scale, decreasing regularization strength at each step. The returned model will have coefficients corresponding to the optimal lambda value as decided during training.
17. Check **Stop early** to end processing when there is no more relative improvement on the training or validation set.

18. Check **Maximum lambdas to search** and enter the maximum number of lambdas to use during the process of lambda search.
19. Check **Maximum active predictors** and enter the maximum number of predictors to use during computation.
This value is used as a stopping criterion to prevent expensive model building with many predictors.
20. Click **OK** to save the model and configuration or continue to the next tab.

Model Output

This tab shows the metrics you are using to assess the fitted model. These fields cannot be edited. The Training column will always contain data. If you selected a train/test split on the Basic Options tab, the Test column will also be filled, unless you have selected an N Fold validation on the Advanced Options tab, in which case the N Fold column will be filled.

After you run your job, the resulting model is stored on the Spectrum Technology Platform server. Click the **Output** button to regenerate the output and click **Model details** to view the entire output in the Machine Learning Model Management tool.

Output Ports

The Logistic Regression stage contains two optional output ports: the Model Score Port and the Model Metrics Port. The functionality of these ports is determined by your selections and input when completing the stage's basic and advanced options. For example, if you choose to conduct N Fold validation by checking the **N Fold** field on the **Advanced Options** tab, the N Fold column in the output metrics generated by the Model Metrics Port will be populated with data. Alternatively, if you choose not to conduct N Fold validation, the N Fold column will be blank. Likewise, The Model Score Port is activated when you check the **Score input data** field on the **Basic Options** tab.

Model Score Port


When you check the **Score input data** field on the **Basic Options** tab, this tells Logistic Regression to calculate predicted values when creating the model, which in turn adds the **Predicted_Value**, **Probability_of_class_A**, and **Probability_of_class_B** columns for that score in the output data. You can attach any kind of sink to this port: a Write to File stage, a Write to Null stage, and so on.

Model Metrics Port

Perform this procedure to use the Model Metrics Port.

The **Model Metrics Port** lets you output the model assessment metrics to a data file. This will help you compare many models generated from within and outside of Spectrum Technology Platform and perform other data processing tasks on the metrics.

1. Open a dataflow that uses the Logistic Regression stage.
2. Attach a Write to File stage or another data output stage to the second output port.
3. Run the job.

- Alternative to step 3 on page 575: Add an inspection point to the channel that connects the Logistic Regression stage to the sink stage you added in step 2 on page 575 by right-clicking the channel and selecting "Add inspection point." Then click the Inspect Current Flow button  on the Enterprise Designer toolbar. Inspection will run and you should see results similar to the ones shown below.

Creation Time	Flow Name	Metrics	Model Name	Model Type	N Fold	Test	Training
10/11/2018 2:37:00 AM	LinearRegressionTest1...	MSE	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10
10/11/2018 2:37:00 AM	LinearRegressionTest1...	RMSE	LinearRegressionTest1...	Linear Regression	0.0707247064967455	0.0567287296394643	1.14687396359675E-05
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Number of observations	LinearRegressionTest1...	Linear Regression	33	17	33
10/11/2018 2:37:00 AM	LinearRegressionTest1...	R2	LinearRegressionTest1...	Linear Regression	-0.366538763835926	0.0268719304981138	0.9999999964065547
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Mean residual deviance	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10

Principal Component Analysis

Introduction

Principal Component Analysis (PCA) is a statistical process that converts a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables known as principal components.

To create your model, you must first complete the **Model Properties** tab. The **Basic Options** and **Advanced Options** tabs provide sufficient default settings to complete a job, but you can change those settings to meet your needs. You then run your job and a limited version of the resulting model appears on the **Model Output** tab; the complete output is available in the Machine Learning Model Management tool. If you are satisfied with the output of your model, you can then expose it and use it in a scoring dataflow.

Defining Model Properties

- Under **Primary Stages > Deployed Stages > Machine Learning**, click the **PCA Options** stage and drag it onto the canvas, placing it where you want on the dataflow and connecting it to other stages.

Note: The input stage must be the data source that contains the principal components for your model. An output stage is not required but you may connect one if you wish to capture your output independent of the Machine Learning Model Management tool.

- Double-click the PCA Options stage to show the **PCA Options** dialog box.
- Enter a **Model name** if you do not want to use the default name.
- Optional: Check the **Overwrite** box to overwrite the existing model with new data.
- Enter the number of **Principal components** you want your model to contain.
- Optional: Enter a **Description** of the model.
- In the **Inputs** table click "Include" for each field whose data you want added to the model.
- Use the **Model Data Type** drop-down to specify whether the input field is to be used as a categorical, datetime, numeric, string, or uniqueid field.
- Click **OK** to save the model and configuration or continue to the next tab.

Configuring Basic Options

1. Configure **Use all factor level**.
 - Leave this option unchecked to skip the first principal component, which has the largest variance in the data.
 - Check this box to retain the first principal component.
2. Select the appropriate **Transform** for the training data.

Transform	Description
Demean	Subtracts the mean of each column.
Descale	Divides by the standard deviation of each column.
None	No transform.
Normalize	Demeans and divides each column by its range (maximum minus minimum).
Standardize	Uses zero mean and unit variance. This is the default transform.

3. Specify how to handle **Missing data** by checking **Skip** or **Impute means**, which will add the mean value for any missing data.
4. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Advanced Options

1. Leave **Ignore constant fields** checked to skip fields that have the same value for each record.
2. Select a **PCA method** from the dropdown list. Note that GLRM and Power are currently experimental.

PCA method	Description
GLRM	Fits a generalized low-rank model with L2 loss function and no regularization; solves for the SVD using local matrix algebra. This option is enabled only if you checked Use all factor level on the Basic Options tab.
GramSVD	Uses a distributed computation of the Gram matrix, followed by a local SVD using the JAMA package.

PCA method	Description
Power	Computes the SVD using the power iteration method.
Randomized	Uses the randomized subspace iteration method.

3. Leave **Maximum iterations** unchecked to have an unlimited number of training iterations (default). Check the box and enter a number to limit the amount of training iterations.
4. Click **OK** to save the model and configuration or continue to the next tab.

Model Output

This tab shows the metrics you are using to assess the fitted model. These fields cannot be edited. After you run your job, the resulting model is stored on the Spectrum Technology Platform server. Click the **Output** button to regenerate the output and click **Model details** to view the entire output in the Machine Learning Model Management tool.


Output Port

The Principal Component Analysis stage contains one optional output port: the Model Metrics Port. This port's functionality is determined by your selections and input when completing the stage's basic and advanced options.

Model Metrics Port

Perform this procedure to use the Model Metrics Port.

The **Model Metrics Port** lets you output the model assessment metrics to a data file. This will help you compare many models generated from within and outside of Spectrum Technology Platform and perform other data processing tasks on the metrics.

1. Open a dataflow that uses the PCA stage.
2. Attach a Write to File stage or another data output stage to the second output port.
3. Run the job.
4. Alternative to step 3 on page 578: Add an inspection point to the channel that connects the PCA stage to the sink stage you added in step 2 on page 578 by right-clicking the channel and selecting "Add inspection point." Then click the Inspect Current Flow button  on the Enterprise Designer toolbar. Inspection will run and you should see results similar to the ones shown below.

Creation Time	Cumulative Proportion	Flow Name	Model Name	Model Type	Principal Component	Proportion of Variance	Standard Deviation
10/11/2018 8:36:00 PM	0.120990707073471	PCA_MODEL_ASSESSMENT	PCA Model Assessment	PCA	PC1	0.120990707073471	1.73278529942543
10/11/2018 8:36:00 PM	0.163608702477588	PCA_MODEL_ASSESSMENT	PCA Model Assessment	PCA	PC2	0.0426179954041175	1.02840755055022
10/11/2018 8:36:00 PM	0.20114715020656	PCA_MODEL_ASSESSMENT	PCA Model Assessment	PCA	PC3	0.0375384477289716	0.965176862701583
10/11/2018 8:36:00 PM	0.236650281720107	PCA_MODEL_ASSESSMENT	PCA Model Assessment	PCA	PC4	0.0355031315135469	0.93864652798561
10/11/2018 8:36:00 PM	0.269754397170741	PCA_MODEL_ASSESSMENT	PCA Model Assessment	PCA	PC5	0.0331041154506347	0.90637880520786

Random Forest Classification

Introduction

Random Forest Classification enables you to perform machine learning by creating models from datasets that use continuous objectives with input variables.

To create your model, you must first complete the Model Properties tab. The Basic Options and Advanced Options tabs provide sufficient default settings to complete a job, but you can change those settings to meet your needs. You then run your job and a limited version of the resulting model appears on the Model Output tab; the complete output is available in the Machine Learning Model Management tool.

Note: For additional information, refer to this article about [Distributed Random Forest \(DRF\)](#) for additional information regarding Random Forest Classification and its options.

Defining Model Properties

1. Under **Primary Stages > Deployed Stages > Machine Learning**, click the **Random Forest Classification** stage and drag it onto the canvas, placing it where you want on the dataflow and connecting it to other stages.

Note: The input stage must be the data source that contains both the objective and input variable fields for your model; an output stage is not required unless you select the Score input data option on the Basic Options tab. You may also connect an output stage if you wish to capture your output independent of the Machine Learning Model Management tool.

2. Double-click the Random Forest Classification stage to show the **Random Forest Classification Options** dialog box.
3. Enter a **Model name** if you do not want to use the default name.
4. Optional: Check the **Overwrite** box to overwrite the existing model with new data.
5. Click the **Objective field** drop-down and select a numeric field.
6. Click **Multinomial levels** and enter the maximum number of categories into which the objective field can be grouped. Note that checking this option will disable the **Score input data** option on the Basic Options tab.
7. Optional: Enter a **Description** of the model.
8. Click **Include** for each field whose data you want added to the model.
Be sure to include the field you selected as the Objective field.
9. Use the **Model Data Type** drop-down to specify whether each input field is to be used as a numeric, categorical, or datetime field.
10. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Basic Options

1. Enter the maximum **Number of trees** in your model.
2. Enter the **Maximum depth**.
This is the maximum number of levels you want your model to contain.
3. Enter the **Minimum rows**.
This is the minimum number of rows (or records) you want your model to contain.
4. Enter the **Number of bins numeric**.
This is the number of bins you want the histogram to build and then split at the best point.
5. Enter the **Number of bins top level**.
This is the minimum number of bins you want at the root level.
6. Enter the **Number of bins categorical**.
This is the maximum number of bins you want the histogram to build and then split at the best point.
7. Check **Sample rate** and enter the percentage of the rows to be used as a sample in each tree.
This can be a value from 0.0 to 1.0.
8. Check **Column sample rate per tree** and enter the column sampling rate for each tree.
This can be a value from 0.0 to 1.0.
9. Check **Columns at each level** and enter the relative change of the column sampling rate for every level.
Valid values range from 1.0 to the number of the selected input predictor. Default is 1.0.
10. Check **Score input data** to add a column for the model prediction (score) to the input data.
Note: This option is disabled if you checked **Multinomial levels** on the Model Properties tab.
11. Specify a value between 1 and 100 as the **Percentage for training data** when the input data is randomly split into training and test data samples.
12. Enter the value of 100 minus the amount you entered in step **11** on page 580 as the **Percentage for test data**.
13. **Seed for sampling** to ensure that when the data is split into test and train data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.
14. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Advanced Options

1. Leave **Ignore constant fields** checked to skip fields that have the same value for each record.

2. Check **Balance classes** to balance the class distribution and either under sample the majority classes or over sample the minority classes.
3. Select a **Histogram type**.

Histogram type	Description
Auto	Buckets are binned from minimum to maximum in steps of $(\text{max-min})/N$. Use this option to specify the type of histogram for finding optimal split points.
QuantilesGlobal	Buckets have equal population. This computes <code>nbins</code> quantiles for each numeric (non-binary) column, then refines/pads each bucket (between two quantiles) uniformly (and randomly for remainders) into a total of <code>nbins_top_level</code> bins.
Random	The algorithm will sample $N-1$ points from minimum to maximum and use the sorted list of those to find the best split.
RoundRobin	The algorithm will cycle through all histogram types (one per tree).
UniformAdaptive	Each feature is binned into buckets of equal step size (not population). This is the quickest method but can lead to less accurate splits if the distribution is highly skewed.

4. Select a **Categorical encoding**.

Categorical encoding	Description
Auto	Automatically performs <code>enum</code> encoding.
Binary	Converts categories to integers, then to binary, and assigns each digit a separate column. Encodes the data in fewer dimensions but with some distortion of the distances. Note: No more than 32 columns can exist per categorical feature.
Eigen	k columns per categorical feature, keeping projections of one-hot-encoded matrix onto k -dim eigen space only.
Enum	Cycles through all histogram types (one per tree).

Categorical encoding	Description
OneHotExplicit	One column exists per category, with "1" or "0" in each cell representing whether the row contains that column's category.

- Leave **Seed for algorithm and N fold** checked and enter a seed number to ensure that when the data is split into test and training data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.
- If you are performing cross-validation, check **N fold** and enter the number of folds.
- If you are performing cross-validation, check **Fold assignment** and select from the dropdown list.

Fold assignment	Description
Auto	Allows the algorithm to automatically choose an option; currently it uses Random.
Modulo	Evenly splits the dataset into the folds and does not depend on the seed.
Random	Randomly splits the data into nfolds pieces; best for large datasets.
Stratified	Stratifies the folds based on the response variable for classification problems. Evenly distributes observations from the different classes to all sets when splitting a dataset into train and test data. This can be useful if there are many classes and the dataset is relatively small.

This field is applicable only if you entered a value in **N fold** and **Fold field** is not specified.

- If you are performing cross-validation, check **Fold field** and select the field that contains the cross-validation fold index assignment from the drop-down list.

This field is applicable only if you did not enter a value in **N fold** and **Fold assignment**.

- Check **Stopping rounds** to end training when the Stopping_metric option does not improve for the specified number of training rounds and enter the number of unsuccessful training rounds to occur before stopping. To disable this feature, specify 0.

The metric is computed on the validation data (if provided); otherwise, training data is used.

- Select a **Stopping metric** to determine when to quit creating new trees.

Stopping metric	Description
AUC	Area under ROC curve.

Stopping metric	Description
	Note: Applicable only to binomial models.
Auto	Defaults to <code>deviance</code> .
Lifftopgroup	Top 1%.
Logloss	Logarithmic loss.
Meanperclasserror	The average misclassification rate.
Misclassification	The value of $(1 - (\text{correct predictions}/\text{total predictions})) * 100$.
MSE	Mean squared error; incorporates both the variance and the bias of the predictor.
RMSE	Root mean square error; measures the differences between values (sample and population values) predicted by a model or an estimator and the values actually observed. Also the square root of MSE.

- Check **Stopping tolerance** and enter a value to specify the relative tolerance for the metric-based stopping to end training if the improvement is less than this value.
This field is enabled only if you checked **Stopping rounds**.
- Check **Minimum split improvement** and enter a value to specify the minimum relative improvement in squared error reduction in order for a split to happen.
When properly executed, this option can help reduce overfitting. Optimal values would be in the 1e-10...1e-3 range. This field is enabled only if you checked **Stopping rounds**.
- Click **OK** to save the model and configuration or continue to the next tab.

Model Output

This tab shows the metrics you are using to assess the fitted model. These fields cannot be edited. The Training column will always contain data. If you selected a training/test split on the **Basic Options** tab, the **Test** column will also be filled, unless you have selected an N Fold validation on the **Advanced Options** tab, in which case the **N Fold** column will be filled.

After you run your job, the resulting model is stored on the Spectrum Technology Platform server. Click the **Output** button to regenerate the output and click **Model details** to view the entire output in the Machine Learning Model Management tool.

Output Ports

The Random Forest Classification stage contains two optional output ports: the Model Score Port and the Model Metrics Port. The functionality of these ports is determined by your selections and input when completing the stage's basic and advanced options. For example, if you choose to conduct N Fold validation by checking the **N Fold** field on the **Advanced Options** tab, the N Fold column in the output metrics generated by the Model Metrics Port will be populated with data. Alternatively, if you choose not to conduct N Fold validation, the N Fold column will be blank. Likewise, The Model Score Port is activated when you check the **Score input data** field on the **Basic Options** tab.

Model Score Port


When you check the **Score input data** field on the **Basic Options** tab, this tells Random Forest Classification to calculate predicted values when creating the model, which in turn adds the **Predicted_Value**, **Probability_of_class_A**, and **Probability_of_class_B** columns for that score in the output data. You can attach any kind of sink to this port: a Write to File stage, a Write to Null stage, and so on.

Note: This port is not functional for Random Forest Classification multinomial models.

Model Metrics Port

Perform this procedure to use the Model Metrics Port.

The **Model Metrics Port** lets you output the model assessment metrics to a data file. This will help you compare many models generated from within and outside of Spectrum Technology Platform and perform other data processing tasks on the metrics.

1. Open a dataflow that uses the Random Forest Classification stage.
2. Attach a Write to File stage or another data output stage to the second output port.
3. Run the job.
4. Alternative to step 3 on page 584: Add an inspection point to the channel that connects the Random Forest Classification stage to the sink stage you added in step 2 on page 584 by right-clicking the channel and selecting "Add inspection point." Then click the Inspect Current Flow button  on the Enterprise Designer toolbar. Inspection will run and you should see results similar to the ones shown below.

Creation Time	Flow Name	Metrics	Model Name	Model Type	N Fold	Test	Training
10/11/2018 2:37:00 AM	LinearRegressionTest1...	MSE	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10
10/11/2018 2:37:00 AM	LinearRegressionTest1...	RMSE	LinearRegressionTest1...	Linear Regression	0.0707247064967455	0.0567287296394643	1.14687396359675E-05
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Number of observations	LinearRegressionTest1...	Linear Regression	33	17	33
10/11/2018 2:37:00 AM	LinearRegressionTest1...	R2	LinearRegressionTest1...	Linear Regression	-0.366538763835926	0.0268719304981138	0.999999964065547
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Mean residual deviance	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10

Random Forest Regression

Introduction

Random Forest Regression enables you to perform machine learning by creating models from datasets that use binary objectives with input variables.

To create your model, you must first complete the **Model Properties** tab. The **Basic Options** and **Advanced Options** tabs provide sufficient default settings to complete a job, but you can change those settings to meet your needs. You then run your job and a limited version of the resulting model appears on the **Model Output** tab; the complete output is available in the Machine Learning Model Management tool.

Note: For more information regarding Random Forest Regression and its options, see [Distributed Random Forest \(DRF\)](#).

Defining Model Properties

1. Under **Primary Stages / Deployed Stages / Machine Learning**, click the **Random Forest Regression** stage and drag it onto the canvas, placing it where you want on the dataflow and connecting it to other stages.

Note: The input stage must be the data source that contains both the objective and input variable fields for your model; an output stage is not required unless you select the Score input data option on the Basic Options tab. You may also connect an output stage if you wish to capture your output independent of the Machine Learning Model Management tool.

2. Double-click the Random Forest Regression stage to show the **Random Forest Regression Options** dialog box.
3. Enter a **Model name** if you do not want to use the default name.
4. Optional: Check the **Overwrite** box to overwrite the existing model with new data.
5. Click the **Objective field** drop-down and select a numeric field.
6. Optional: Enter a **Description** of the model.
7. Click **Include** for each field whose data you want added to the model; be sure to include the field you selected as the Objective field.
8. Use the **Model Data Type** drop-down to specify whether each input field is to be used as a numeric, categorical, or datetime field.
9. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Basic Options

1. Enter the maximum **Number of trees** in your model. Default is 50.
2. Enter the **Maximum depth**.

This is the maximum number of levels you want your model to contain. The default is 5.

3. Enter the **Minimum rows**.
This is the minimum number of rows (or records) you want your model to contain. The default is 10.
4. Enter the **Number of bins numeric**.
This is the number of bins you want the histogram to build and then split at the best point. The default is 20.
5. Enter the **Number of bins top level**.
This is the minimum number of bins you want at the root level. The default is 1024.
6. Enter the **Number of bins categorical**.
This is the maximum number of bins you want the histogram to build and then split at the best point. The default is 1024.
7. Check **Sample rate** and enter the percentage of the rows to be used as a sample in each tree. This can be a value from 0.0 to 1.0.
8. Check **Column sample rate per tree** and enter the column sampling rate for each tree. This can be a value from 0.0 to 1.0.
9. Check **Columns at each level** and enter the relative change of the column sampling rate for every level.
This option defaults to 1.0 and can be a value from 0.0 to 2.0.
10. Check **Score input data** to add a column for the model prediction (score) to the input data.
11. Specify a value between 1 and 100 as the **Percentage for training data** when the input data is randomly split into training and test data samples.
12. Enter the value of 100 minus the amount you entered in step **11** on page 586 as the **Percentage for test data**.
13. **Seed for sampling** to ensure that when the data is split into test and train data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.
14. Click **OK** to save the model and configuration or continue to the next tab.

Configuring Advanced Options

1. Leave **Ignore constant fields** checked to skip fields that have the same value for each record.
2. Select a **Histogram type**.

Histogram type	Description
Auto	Buckets are binned from minimum to maximum in steps of $(\text{max}-\text{min})/N$. Use this option to specify the type of histogram for finding optimal split points.

Histogram type	Description
QuantilesGlobal	Buckets have equal population. This computes <code>nbins</code> quantiles for each numeric (non-binary) column, then refines/pads each bucket (between two quantiles) uniformly (and randomly for remainders) into a total of <code>nbins_top_level</code> bins.
Random	The algorithm will sample $N-1$ points from minimum to maximum and use the sorted list of those to find the best split.
RoundRobin	The algorithm will cycle through all histogram types (one per tree).
UniformAdaptive	Each feature is binned into buckets of equal step size (not population). This is the quickest method but can lead to less accurate splits if the distribution is highly skewed.

3. Select a **Categorical encoding**.

Categorical encoding	Description
Auto	Automatically performs <code>enum</code> encoding.
Binary	Converts categories to integers, then to binary, and assigns each digit a separate column. Encodes the data in fewer dimensions but with some distortion of the distances. Note: No more than 32 columns can exist per categorical feature.
Eigen	k columns per categorical feature, keeping projections of one-hot-encoded matrix onto k -dim eigen space only.
Enum	Cycles through all histogram types (one per tree).
OneHotExplicit	One column exists per category, with "1" or "0" in each cell representing whether the row contains that column's category.

4. Leave **Seed for algorithm and N fold** checked and enter a seed number to ensure that when the data is split into test and training data it will occur the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.

5. Check **N fold** and enter the number of folds if you are performing cross-validation.
6. If you are performing cross-validation, check **Fold assignment** and select from the dropdown list .

Fold assignment	Description
Auto	Allows the algorithm to automatically choose an option; currently it uses Random.
Modulo	Evenly splits the dataset into the folds and does not depend on the seed.
Random	Randomly splits the data into nfolds pieces; best for large datasets.

This field is applicable only if you entered a value in **N fold** and **Fold field** is not specified.

7. If you are performing cross-validation, check **Fold field** and select the field that contains the cross-validation fold index assignment from the drop-down list.

This field is applicable only if you did not enter a value in **N fold** and **Fold assignment**.

8. Check **Stopping rounds** to end training when the Stopping_metric option does not improve for the specified number of training rounds and enter the number of unsuccessful training rounds to occur before stopping. To disable this feature, specify 0.

The metric is computed on the validation data (if provided); otherwise, training data is used.

9. Select a **Stopping metric** to determine when to quit creating new trees.

Stopping metric	Description
Auto	Defaults to <code>deviance</code> .
deviance	Mean residual deviance; identical to MSE.
MAE	Mean absolute error; the difference between two continuous variables.
MSE	Mean squared error; incorporates both the variance and the bias of the predictor.
RMSE	Root mean square error; measures the differences between values (sample and population values) predicted by a model or an estimator and the values actually observed. Also the square root of MSE.
RMSLE	Root mean squared logarithmic error; measures the ratio between predicted and actual.

10. Check **Stopping tolerance** and enter a value to specify the relative tolerance for the metric-based stopping to end training if the improvement is less than this value.
11. Check **Minimum split improvement** and enter a value to specify the minimum relative improvement in squared error reduction in order for a split to happen.
When properly executed, this option can help reduce overfitting. Optimal values would be in the 1e-10...1e-3 range. This field is enabled only if you checked **Stopping rounds**.
12. Click **OK** to save the model and configuration or continue to the next tab.

Model Output

This tab shows the metrics you are using to assess the fitted model. These fields cannot be edited. The Training column will always contain data. If you selected a training/test split on the Basic Options tab, the Test column will also be filled, unless you have selected an N Fold validation on the Advanced Options tab, in which case the N Fold column will be filled.

After you run your job, the resulting model is stored on the Spectrum Technology Platform server. Click the **Output** button to regenerate the output and click **Model details** to view the entire output in the Machine Learning Model Management tool.

Output Ports

The Random Forest Regression stage contains two optional output ports: the Model Score Port and the Model Metrics Port. The functionality of these ports is determined by your selections and input when completing the stage's basic and advanced options. For example, if you choose to conduct N Fold validation by checking the **N Fold** field on the **Advanced Options** tab, the N Fold column in the output metrics generated by the Model Metrics Port will be populated with data. Alternatively, if you choose not to conduct N Fold validation, the N Fold column will be blank. Likewise, The Model Score Port is activated when you check the **Score input data** field on the **Basic Options** tab.

Model Score Port


When you check the **Score input data** field on the **Basic Options** tab, this tells Random Forest Regression to calculate predicted values when creating the model, which in turn adds the **Predicted_Value** column for that score in the output data. You can attach any kind of sink to this port: a Write to File stage, a Write to Null stage, and so on.

Model Metrics Port

Perform this procedure to use the Model Metrics Port.

The **Model Metrics Port** lets you output the model assessment metrics to a data file. This will help you compare many models generated from within and outside of Spectrum Technology Platform and perform other data processing tasks on the metrics.

1. Open a dataflow that uses the Random Forest Regression stage.
2. Attach a Write to File stage or another data output stage to the second output port.
3. Run the job.

- Alternative to step 3 on page 589: Add an inspection point to the channel that connects the Random Forest Regression stage to the sink stage you added in step 2 on page 589 by right-clicking the channel and selecting "Add inspection point." Then click the Inspect Current Flow button  on the Enterprise Designer toolbar. Inspection will run and you should see results similar to the ones shown below.

Creation Time	Flow Name	Metrics	Model Name	Model Type	N Fold	Test	Training
10/11/2018 2:37:00 AM	LinearRegressionTest1...	MSE	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10
10/11/2018 2:37:00 AM	LinearRegressionTest1...	RMSE	LinearRegressionTest1...	Linear Regression	0.0707247064967455	0.0567287296394643	1.14687396359675E-05
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Number of observations	LinearRegressionTest1...	Linear Regression	33	17	33
10/11/2018 2:37:00 AM	LinearRegressionTest1...	R2	LinearRegressionTest1...	Linear Regression	-0.366538763835926	0.0268719304981138	0.999999964065547
10/11/2018 2:37:00 AM	LinearRegressionTest1...	Mean residual deviance	LinearRegressionTest1...	Linear Regression	0.0050019841090508	0.00321814876650744	1.31531988837612E-10

Universal Addressing Stages

Auto Complete Loqate

Auto Complete Loqate offers real-time entry of address data for fast, accurate results. Users are returned instant results based on each character entered into the form, ensuring only accurate data is entered into the database. Auto Complete Loqate also includes the Powersearch option, which reduces input time by up to 80% for 238 countries by using data in the form of an index file.

Input

The following table lists the input for Auto Complete Loqate.

Table 73: Input Format

Field Name	Description
AddressLine1	The first address line.
AddressLine2	The second address line.
AddressLine3	The third address line.
AddressLine4	The fourth address line.
City	The city name.

Field Name	Description
Country	<p>The country code or name, in any of the following formats:</p> <ul style="list-style-type: none"> • 2-digit ISO country code • 3-digit UPU Country code • English country name <p>For a list of ISO codes, see ISO Country Codes and Module Support.</p>
FirmName	The company or firm name.
PostalCode	The postal code for the address.
StateProvince	The state or province.

Options

Table 74: Auto Complete Loqate Options

Option Name	Description
Database	Specifies the database to be used for address processing. Only databases that have been defined in the Database Resources panel in the Management Console are available.
Casing	<p>Specifies the casing of the output data. One of the following:</p> <p>Mixed Returns the output in mixed case (default). For example:</p> <p>123 Main St Mytown FL 12345</p> <p>Upper Returns the output in upper case. For example:</p> <p>123 MAIN ST MYTOWN FL 12345</p>

Option Name	Description
Default country	Specifies the default country. You should specify the country where most of your addresses reside. For example, if most of the addresses you process are in Germany, specify Germany.
Country format	<p>Specifies the format to use for the country name returned in the Country output field. For example, if you select English, the country name "Deutschland" would be returned as "Germany".</p> <p>English Names Use English country names (default).</p> <p>ISO Codes Use two-letter ISO abbreviation for the countries instead of country names.</p> <p>UPU Codes Use Universal Postal Union abbreviation for the countries instead of country names.</p>
Script/Alphabet	<p>Specifies the alphabet or script in which the output should be returned. This option is bi-directional and generally takes place from Native to Latin and Latin to Native.</p> <p>Input Script Do not perform transliteration and provide output in the same script as the input (default).</p> <p>Native Output in the native script for the selected country wherever possible.</p> <p>Latin (English) Use English values.</p>
Maximum records to return	The maximum number of addresses that Auto Complete Loqate should return. The default is 10.

Option Name	Description
Prefer Powersearch	<p>Reduces input time by up to 80% for 240 countries by using data in the form of an index file. When you conduct a search, the Loqate Engine will first look for the corresponding index. If present, the method will attempt to instantly return a list of candidate addresses. If the index is not present, or if the index does not return any results, the original search process will be triggered.</p> <p>Note: Powersearch can be performed when there are two and only two fields in the input file: the Country field and any one of the AddressLine fields. If you select this option and your input file contains additional fields, the original search process will automatically be triggered.</p> <p>To conduct its search, Auto Complete indexes use up to the first 10 characters for searches within the United States and up to the first 15 characters for searches within all other eligible countries. Spaces and punctuation are not factored into this count.</p> <p>Powersearch cannot be used for the following countries: Botswana, Ethiopia, India, Kazakhstan, Malaysia, Mongolia, Saint Kitts and Nevis, and San Marino.</p> <p>Note: You must have a valid license for Powersearch processing. If you enable Powersearch processing but are not licensed for this feature, or if your license has expired, your entire job will fail.</p>
Duplicate handling	<p>Enables the duplicate handling mask and specifies how duplicate records are processed and removed. Select one or more of the following options:</p> <p>Single Selected by default. Pre-process the input and remove duplicates that occur in a single field.</p> <p>Multi Selected by default. Preprocess the input and remove duplicates across all fields.</p> <p>Non-standard Preprocess the input and remove duplicates in fields that are not standard address fields.</p> <p>Output Selected by default. Post-process the output from verification and remove duplicates from non-verified fields.</p>
Data license error handling	<p>Specifies how you want Spectrum Technology Platform to respond when a data license error occurs.</p> <p>Fail the job Fail the entire job if a data license error occurs.</p> <p>Fail the record Fail the record(s) for which the data license error occurs and continue processing.</p>

Output

The output from Auto Complete Loqate is optional and corresponds directly to the fields you selected in the Output Fields section of the Auto Complete Loqate Options dialog box.

Table 75: Auto Complete Loqate Output

Field Name	Description
AddressLine1	The first address line.
AddressLine2	The second address line.
AddressLine3	The third address line.
AddressLine4	The fourth address line.
City	The city name.
Country	The three-character ISO 3166-1 Alpha 3 code for the country. For a list of ISO codes, see ISO Country Codes and Module Support .
FirmName	The firm name.
HouseNumber	The ending house number for the range in which the candidate address's house number falls.
PostalCode	The postal code.
PostalCode.AddOn	The last four digits of the ZIP + 4 [®] Code.
ProcessedBy	Indicates which address coder processed the address. LOQATE The Loqate coder processed the address.

Field Name	Description				
StateProvince	The validated state/province or its abbreviated value.				
Status	Reports the success or failure of the match attempt. <table border="0"> <tr> <td>null</td> <td>Success</td> </tr> <tr> <td>F</td> <td>Failure</td> </tr> </table>	null	Success	F	Failure
null	Success				
F	Failure				
Status.Code	The reason for failure, if there is one. <ul style="list-style-type: none"> • DisabledCoder • RequestFailed • NoLookupAddressFound 				
Status.Description	A description of the problem, if there is one. <table border="0"> <tr> <td>Did not return multiples</td> <td>The input address matched only one address in the database. Auto Complete Loqate returns data only if multiple possible matches were found.</td> </tr> <tr> <td>Not able to look up the address pattern</td> <td>Auto Complete Loqate is not able to process the partial address.</td> </tr> </table>	Did not return multiples	The input address matched only one address in the database. Auto Complete Loqate returns data only if multiple possible matches were found.	Not able to look up the address pattern	Auto Complete Loqate is not able to process the partial address.
Did not return multiples	The input address matched only one address in the database. Auto Complete Loqate returns data only if multiple possible matches were found.				
Not able to look up the address pattern	Auto Complete Loqate is not able to process the partial address.				

Get Candidate Addresses

Get Candidate Addresses returns a list of addresses that are considered matches for a given input address. Get Candidate Addresses returns candidate addresses only if the input address matches multiple addresses in the postal database. If the input address matches only one address in the postal database, then no address data is returned.

For addresses outside the U.S. and Canada, you may notice inconsistent results between the multiple matches returned by Validate Address and the results for that same address returned by Get Candidate Addresses. If you experience inconsistent results, it is likely because you set the performance tuning setting in Validate Address to a value other than 100. To obtain consistent results between Get Candidate Addresses and Validate Address, set the performance tuning option to 100.

Note: By default, Get Candidate Addresses does not match to individual house numbers. Rather, it uses house number ranges for each street. After Get Candidate Addresses has determined the street name, city name, state/province name, and postal code, it checks to make sure the input house number falls within one of the ranges of house numbers given for the matched street name. The same type of logic applies to unit numbers. If you want to determine that an individual house number is valid, you should use the Validate Address

Delivery Point Validation (DPV) processing option. DPV processing is only available for U.S. addresses.

The Canadian coder contains a reverse lookup routine that takes as input a specific postal code and returns the street information stored in the database for that postal code. To use this function enter nothing but a Canadian postal code in the PostalCode field. See the second example to view the return from a sample postal code.

Get Candidate Addresses is part of Spectrum Universal Address.

U.S. Address Example

AddressLine1:

PO Box 1 City: NY State: NY

Preview Output:

Field Name	Output Record 1	Output Record 2
AddressLine1	PO Box 1	PO Box 1
City	New York	New York
Country	USA	USA
HouseNumberHigh	60	9
HouseNumberLow	1	1
HouseNumberParity	B	B
MatchLevel	A	A
PostalCode	10002	10008
PostalCode.AddOn	0001	0001
ProcessedBy	USA	USA
RecordType	PostOfficeBox	PostOfficeBox
RecordType.Default		

Field Name	Output Record 1	Output Record 2
StateProvince	NY	NY
UnitNumberParity		

Canadian Address Example

PostalCode:

A1A1A1

Preview Output:

Field Name	Output Record 1	Output Record 2
AddressLine1	LOWER BATTERY RD	LOWER BATTERY RD
City	ST. JOHN'S	ST. JOHN'S
Country	CAN	CAN
HouseNumberHigh	000003	000004 A
HouseNumberLow	000001	000002
HouseNumberParity	O	E
MatchLevel	A	A
PostalCode	A1A1A1	A1A1A1
ProcessedBy	CAN	CAN
RecordType	Normal	Normal
StateProvince	NL	NL

Input

The following table lists the input for Get Candidate Addresses.

Table 76: Input Format

Field Name	Description
AddressLine1	The first address line.
AddressLine2	The second address line.
AddressLine3	The third address line. Does not apply to U.S. and Canadian addresses.
AddressLine4	The fourth address line. Does not apply to U.S. and Canadian addresses.
AddressLine5	The fifth address line. Applies only to U.K. addresses. May contain street name, unit number, building number, and so on.
City	The city name.
StateProvince	The state or province. For U.S. addresses only, you may put the state in the City field instead of the StateProvince field.
PostalCode	The postal code for the address. For U.S. addresses this is the ZIP Code™ in one of the following formats: 99999 99999-9999 A9A9A9 A9A 9A9 9999 999 Note: For Canadian addresses you can complete just this field and have candidate address data returned. For other countries, AddressLine1 and AddressLine2 must also be completed.

Field Name	Description
Country	<p>The country code or name, in any of the following formats:</p> <ul style="list-style-type: none"> • 2-digit ISO country code • 3-digit UPU Country code • English country name • French country name • German country name • Spanish country name <p>For a list of ISO codes, see ISO Country Codes and Module Support.</p>
FirmName	The company or firm name.
USUrbanName	U.S. address urbanization name. Used primarily for Puerto Rico addresses.

Options

Table 77: Get Candidate Addresses Options

Option Name	Description
Enable U.S. address processing	<p>Specifies whether or not to process U.S. addresses. If you enable U.S. address processing Get Candidate Addresses will attempt to retrieve candidate addresses for U.S. addresses. If you disable U.S. address processing, U.S. addresses will fail, meaning they are returned with an "F" in the Status output field. The output field Status.Code will say "DisabledCoder." If you are not licensed for U.S. address processing you must disable U.S. address processing in order for your jobs to complete successfully, regardless of whether or not they contain U.S. addresses.</p> <p>Note: You must have a valid license for U.S. address processing to successfully process U.S. addresses. If you enable U.S. address processing but are not licensed for this feature, or your license has expired, your entire job will fail.</p>
Database	Specifies the database to be used for U.S. address processing. Only databases that have been defined in the US Database Resources panel in the Management Console are available.

Option Name	Description
Enable Canadian address processing	<p>Specifies whether or not to process Canadian addresses. If you enable Canadian address processing Get Candidate Addresses will attempt to retrieve candidate addresses for Canadian addresses. If you disable Canadian address processing, Canadian addresses will fail, meaning they are returned with an "F" in the Status output field. The output field Status.Code will say "DisabledCoder." If you are not licensed for Canadian address processing you must disable Canadian address processing in order for your jobs to complete successfully, regardless of whether or not they contain Canadian addresses.</p> <p>Note: You must have a valid license for Canadian address processing to successfully process Canadian addresses. If you enable Canadian address processing but are not licensed for this feature, or your license has expired, your entire job will fail.</p>
Database	<p>Specifies the database to be used for Canadian address processing. Only databases that have been defined in the Canadian Database Resources panel in the Management Console are available.</p>
Enable International address processing	<p>Specifies whether or not to process international addresses (addresses outside the U.S. and Canada). If you enable international address processing Get Candidate Addresses will attempt to retrieve candidate addresses for international addresses. If you disable international address processing, international addresses will fail, meaning they are returned with an "F" in the Status output field. The output field Status.Code will say "DisabledCoder." If you are not licensed for international address processing you must disable international address processing in order for your jobs to complete successfully, regardless of whether or not they contain international addresses.</p> <p>Note: You must have a valid license for international address processing to successfully process international addresses. If you enable international address processing but are not licensed for this feature, or your license has expired, your entire job will fail.</p>
Database	<p>Specifies the database to be used for international address processing. Only databases that have been defined in the International Database Resources panel in the Management Console are available.</p>

Option Name	Description
Casing	<p>Specifies the casing of the output data. One of the following:</p> <p>Mixed Returns the output in mixed case (default). For example:</p> <pre>123 Main St Mytown FL 12345</pre> <p>Upper Returns the output in upper case. For example:</p> <pre>123 MAIN ST MYTOWN FL 12345</pre>
Maximum records to return	The maximum number of candidate addresses that Get Candidate Addresses should return. The default is 10. The maximum is 10.
Return short city name	For U.S. addresses, specifies whether or not to return the USPS®-approved abbreviation for the city, if there is one. The USPS® provides abbreviations for city names that are 14 characters long or longer. City abbreviations are 13 characters or less and can be used when there is limited space on the mailing label. If there is no short city name for the city, then the full city name is returned.
Dual address match logic	<p>(U.S. addresses only). Controls whether Get Candidate Addresses should return a street match or a PO Box/Rural Route/Highway Contract match when the address contains both street and PO Box/Rural Route/Highway Contract information. For more information, see About Dual Address Logic on page 636.</p> <p>Normal Match (Default) USPS® CASS™ regulations determine the address returned based on the following order of priority:</p> <ol style="list-style-type: none"> 1. PO Box 2. Firm 3. Highrise 4. Street 5. Rural Route 6. General Delivery <p>Street Match Return a street match, regardless of the address line.</p> <p>PO Box Match Return a PO Box match, regardless of the address line.</p>

Option Name	Description
Street matching	The strictness of the street name match (U.S. addresses only).
	Exact The input street name must match the database exactly.
	Tight The matching algorithm is "tight."
	Medium The matching algorithm is "medium" (default).
	Loose The matching algorithm is "loose."
Firm matching	The strictness of the firm name match (U.S. addresses only).
	Exact The input firm name must match the database exactly.
	Tight The matching algorithm is "tight."
	Medium The matching algorithm is "medium" (default).
	Loose The matching algorithm is "loose."
Directional matching	The strictness of the directional match.
	Exact The input directional must match the database exactly.
	Tight The matching algorithm is "tight."
	Medium The matching algorithm is "medium" (default).
	Loose The matching algorithm is "loose."
Perform enhanced street matching	Specifies whether or not to perform Enhanced Street Matching (ESM). ESM applies extra matching logic with additional data to any input address that is not matched through the regular address validation process. ESM applies to U.S. addresses only.

Option Name	Description
Search address lines on fail	<p>Specifies whether ValidateAddress will search address lines for the city, state/province, and postal code.</p> <p>This option enables Validate Address to search the AddressLine input fields for the city, state/province, postal code, and country when the address cannot be matched using the values in the City, StateProvince, and PostalCode input fields.</p> <p>Consider enabling this option if your input addresses have the city, state/province, and postal code information in the AddressLine fields.</p> <p>Consider disabling this option if your input addresses use the City, State/Province and PostalCode fields. If you enable this option and these fields are used, there is an increased possibility that Validate Address will fail to correct values in these fields (for example a misspelled city name).</p>

Output

Get Candidate Addresses returns the following output.

Table 78: Get Candidate Addresses Output

Field Name	Description
AddressLine1	The first address line.
AddressLine2	The second address line.
AddressLine3	The third address line.
AddressLine4	The fourth address line.
AddressLine5	For U.K. addresses only. If the address was validated, the fifth line of the validated and standardized address. If the address could not be validated, the fifth line of the input address without any changes.

Field Name	Description
CanadianDeliveryInstallationAreaName	Delivery installation name (Canadian addresses only)
CanadianDeliveryInstallationQualifierName	Delivery installation qualifier (Canadian addresses only)
CanadianDeliveryInstallationType	Delivery installation type (Canadian addresses only)
City	The city name.
Country	The three-character ISO 3166-1 Alpha 3 code for the country. For a list of ISO codes, see ISO Country Codes and Module Support .
FirmName	The firm name.
HouseNumberHigh	The ending house number for the range in which the candidate address's house number falls.
HouseNumberLow	The beginning house number for the range in which the candidate address's house number falls.
HouseNumberParity	Indicates the numbering scheme for the house numbers between HouseNumberLow and HouseNumberHigh, as follows: <ul style="list-style-type: none"> E Only even values O Only odd values B Both

Field Name	Description
MatchLevel	<p>For addresses outside the U.S. and Canada, identifies the match level for the candidate address. U.S. and Canadian addresses are always "A." One of the following:</p> <p>A The candidate matches the input address at the street level.</p> <p>B The candidate matches the input address at the state/province level.</p>
PostalCode	The postal code. In the U.S. this is the ZIP Code™.
PostalCode.AddOn	The last four digits of the ZIP + 4® Code. U.S. addresses only.
RecordType	<p>The type of address record, as defined by U.S. and Canadian postal authorities (U.S. and Canadian addresses only):</p> <ul style="list-style-type: none"> • FirmRecord • GeneralDelivery • HighRise • PostOfficeBox • RRHighwayContract • Normal
RecordType.Default	<p>Code indicating the "default" match:</p> <p>Y The address matches a default record.</p> <p>null The address does not match a default record.</p>
StateProvince	The validated state/province or its abbreviated value.
Status	<p>Reports the success or failure of the match attempt.</p> <p>null Success</p> <p>F Failure</p>
Status.Code	<p>The reason for failure, if there is one. There is only one possible value:</p> <ul style="list-style-type: none"> • DisabledCoder • RequestFailed

Field Name	Description
Status.Description	<p>A description of the problem, if there is one.</p> <p>Did not return multiples The input address matched only one address in the database. Get Candidate Addresses only returns data if multiple possible matches were found.</p> <p>Number of candidates is not greater than 1 The input address matched more than one address in the database but no addresses were returned.</p> <p>PerformUSProcessing disabled This value will appear if Status.Code=DisabledCoder.</p> <p>PerformCanadianProcessing disabled This value will appear if Status.Code=DisabledCoder.</p> <p>PerformInternationalProcessing disabled This value will appear if Status.Code=DisabledCoder.</p>
UnitNumberHigh	The ending unit number for the range in which the candidate address's unit number falls.
UnitNumberLow	The beginning unit number for the range in which the candidate address's unit number falls.
UnitNumberParity	<p>Indicates the numbering scheme for the unit numbers between UnitNumberLow and UnitNumberHigh, as follows:</p> <p>E Only even values</p> <p>O Only odd values</p> <p>B Both</p>
USUrbanName	The validated city urbanization name. Urbanization names are used primarily for Puerto Rico addresses.

Get Candidate Addresses Loqate

Get Candidate Addresses Loqate returns a list of addresses that are considered matches for a given input address. Get Candidate Addresses Loqate returns candidate addresses only if the input address matches multiple addresses in the postal database. If the input address matches only one address

in the postal database, then no address data is returned. The Country input field is required; if this field is blank, no output will be returned.

Note: By default, Get Candidate Addresses Loqate does not match to individual house numbers. Rather, it uses house number ranges for each street. After Get Candidate Addresses Loqate has determined the street name, city name, state/province name, and postal code, it checks to make sure the input house number falls within one of the ranges of house numbers given for the matched street name. The same type of logic applies to unit numbers.

Get Candidate Addresses Loqate is part of Spectrum Universal Address.

U.S. Address Example

Preview Input:

Field Name	Input
AddressLine1	PO Box 1
AddressLine2	73 baruch
AddressLine3	
AddressLine4	
City	ny
StateProvince	ny
PostalCode	
Country	usa
FirmName	

U.S. Address Example

Preview Output:

Field Name	Output Record 1	Output Record 2	Output Record 3	Output Record 4	Output Record 5
AddressLine1	PO Box 1 73	PO Box 1 73	PO Box 1 73	PO Box 1 73	PO Box 1 73
AddressLine2	Baruch	Baruch	Baruch	Baruch	Baruch
AddressLine3					
AddressLine4					
City	New York	New York	New York	New York	New York
Country	USA	USA	USA	USA	USA
FirmName					
PostalCode	10002	10008	10009	10012-0003	10013
PostalCode.AddOn				0003	
ProcessedBy	LOQATE	LOQATE	LOQATE	LOQATE	LOQATE
StateProvince	NY	NY	NY	NY	NY

Canadian Address Example

PostalCode:

A1A1A1

Preview Output:

Field Name	Output Record 1	Output Record 2
AddressLine1	LOWER BATTERY RD	LOWER BATTERY RD
City	ST. JOHN'S	ST. JOHN'S
Country	CAN	CAN

Field Name	Output Record 1	Output Record 2
HouseNumberHigh	000003	000004 A
HouseNumberLow	000001	000002
HouseNumberParity	O	E
MatchLevel	A	A
PostalCode	A1A1A1	A1A1A1
ProcessedBy	CAN	CAN
RecordType	Normal	Normal
StateProvince	NL	NL

Input

The following table lists the input for Get Candidate Addresses Loqate.

Table 79: Input Format

Field Name	Description
AddressLine1	The first address line.
AddressLine2	The second address line.
AddressLine3	The third address line.
AddressLine4	The fourth address line.
City	The city name.

Field Name	Description
Country	<p>The country code or name, in any of the following formats:</p> <ul style="list-style-type: none"> • 2-digit ISO country code • 3-digit UPU Country code • English country name <p>For a list of ISO codes, see ISO Country Codes and Module Support.</p> <p>Note: This field is required. If this field is blank, no output will be returned.</p>
FirmName	The company or firm name.
PostalCode	The postal code for the address. For U.S. addresses this is the ZIP Code™ in one of the following formats:
StateProvince	<p>The state or province.</p> <p>For U.S. addresses only, you may put the state in the City field instead of the StateProvince field.</p>

Options

Table 80: Get Candidate Addresses Loqate Options

Option Name	Description
Database	Specifies the database to be used for address processing. Only databases that have been defined in the Management Console are available.

Option Name	Description
Casing	<p>Specifies the casing of the output data. One of the following:</p> <p>Mixed Returns the output in mixed case (default). For example:</p> <p>123 Main St Mytown FL 12345</p> <p>Upper Returns the output in upper case. For example:</p> <p>123 MAIN ST MYTOWN FL 12345</p>
Address Lookup Process	<p>Specifies the method of searching for candidates. One of the following:</p> <p>Search Enter a full or partial address as input and return as output a list of closely matching results (default).</p> <p>Verify Enter address information in address lines, address components, or a combination of both as input and return as output results that more closely match the input.</p>
Default Country	<p>Specifies the default country. You should specify the country where most of your addresses reside. For example, if most of the addresses you process are in Germany, specify Germany. Get Candidate Address Loqate uses the country you specify to attempt validation when it cannot determine the country from the StateProvince, PostalCode, and Country address fields.</p>
Country format	<p>Specifies the format to use for the country name returned in the Country output field. For example, if you select English, the country name "Deutschland" would be returned as "Germany".</p> <p>English Names Use English country names (default).</p> <p>ISO Codes Use two-letter ISO abbreviation for the countries instead of country names.</p> <p>UPU Codes Use Universal Postal Union abbreviation for the countries instead of country names.</p>

Option Name	Description						
Script/Alphabet	Specifies the alphabet or script in which the output should be returned. This option is bi-directional and generally takes place from Native to Latin and Latin to Native. <table border="0" data-bbox="548 449 1433 642"> <tr> <td>Input Script</td> <td>Do not perform transliteration and provide output in the same script as the input (default).</td> </tr> <tr> <td>Native</td> <td>Output in the native script for the selected country wherever possible.</td> </tr> <tr> <td>Latin (English)</td> <td>Use English values.</td> </tr> </table>	Input Script	Do not perform transliteration and provide output in the same script as the input (default).	Native	Output in the native script for the selected country wherever possible.	Latin (English)	Use English values.
Input Script	Do not perform transliteration and provide output in the same script as the input (default).						
Native	Output in the native script for the selected country wherever possible.						
Latin (English)	Use English values.						
Maximum records to return	The maximum number of candidate addresses that Get Candidate Addresses Loqate should return. The default is 10. The maximum is 99.						

Output

Get Candidate Addresses Loqate returns the following output.

Table 81: Get Candidate Addresses Loqate Output

Field Name	Description
AddressLine1	The first address line.
AddressLine2	The second address line.
AddressLine3	The third address line.
AddressLine4	The fourth address line.
City	The city name.
Country	The three-character ISO 3166-1 Alpha 3 code for the country. For a list of ISO codes, see ISO Country Codes and Module Support .

Field Name	Description
FirmName	The firm name.
PostalCode	The postal code. In the U.S. this is the ZIP Code™.
PostalCode.AddOn	The last four digits of the ZIP + 4® Code. U.S. addresses only.
ProcessedBy	Indicates which address coder processed the address. LOQATE The Loqate coder processed the address.
StateProvince	The validated state/province or its abbreviated value.
Status	Reports the success or failure of the match attempt. null Success F Failure
Status.Code	The reason for failure, if there is one. There is only one possible value: • RequestFailed
Status.Description	A description of the problem, if there is one. There is only one possible value: Did not return multiples The input address matched only one address in the database. Get Candidate Addresses Loqate only returns data if multiple possible matches were found.

Get City State Province

Get City State Province returns a city and state/province for a given input postal code.

Note: Get City State Province works with U.S. and Canadian addresses only.

Get City State Province is part of Spectrum Universal Address.

Input

The following table shows the input fields.

Table 82: Get City State Province Input

Field Name	Description
PostalCode	A U.S. ZIP Code™ or Canadian postal code in one of the following formats: 99999 99999-9999 A9A9A9 A9A 9A9

Options

Table 83: Get City State Province Options

Option Name	Description
Enable U.S. address processing	Specifies whether or not to process U.S. addresses. If you enable U.S. address processing Get City State Province will attempt to return the state for U.S. addresses. If you disable U.S. address processing, U.S. addresses will fail, meaning they are returned with an "F" in the Status output field. The output field Status.Code will say "DisabledCoder." If you are not licensed for U.S. address processing you must disable U.S. address processing in order for your jobs to complete successfully, regardless of whether or not they contain U.S. addresses. Note: You must have a valid license for U.S. address processing to successfully process U.S. addresses.
Database	Specifies the database to be used for U.S. address processing. Only databases that have been defined in the US Database Resources panel in the Management Console are available.

Option Name	Description
Enable Canadian address processing	<p>Specifies whether or not to process Canadian addresses. If you enable Canadian address processing Get City State Province will attempt to return the province for Canadian addresses. If you disable Canadian address processing, Canadian addresses will fail, meaning they are returned with an "F" in the Status output field. The output field Status.Code will say "DisabledCoder." If you are not licensed for Canadian address processing you must disable Canadian address processing in order for your jobs to complete successfully, regardless of whether or not they contain Canadian addresses.</p> <p>Note: You must have a valid license for Canadian address processing to successfully process Canadian addresses.</p>
Database	<p>Specifies the database to be used for Canadian address processing. Only databases that have been defined in the Canadian Database Resources panel in the Management Console are available.</p>
Include non-mailing city	<p>Specifies whether or not to include non-mailing city names in the output. A non-mailing city name is an alternate name for the primary city name. For example, Hollywood is a non-mailing city name for Los Angeles.</p>
Maximum records to return	<p>Specifies the maximum number of city-state/province pairs to return. The default value is 10.</p>

Output

Get City State Province returns the matching city and state/province for the input postal code as well as a code to indicate the success or failure of the match attempt. If more than one city/state or city/province matches the input postal code, multiple output records are returned.

Table 84: Get City State Province Output

Field Name	Description
City	The matched city name.

Field Name	Description
City.Type	<p>The USPS® standardized city name type (U.S. addresses only).</p> <p>V Vanity (non-mailing) city name.</p> <p>P Primary. The city name is the primary mailing city name.</p> <p>S Secondary. The city name is an alternate city name but is acceptable. A city can have multiple secondary city names.</p>
PostalCode	The input postal code.
ProcessedBy	<p>Indicates which address coder processed the address. One of the following:</p> <p>USA The U.S. address coder processed the address.</p> <p>CAN The Canadian address coder processed the address.</p>
StateProvince	The validated state/province or its abbreviated value.
Status	<p>Reports the success or failure of the match attempt.</p> <p>null Success</p> <p>F Failure</p>
Status.Code	<p>The reason for failure, if there is one. The only valid value is:</p> <ul style="list-style-type: none"> • DisabledCoder • UnrecognizedPostalCode
Status.Description	<p>The description of the failure. The valid values are:</p> <p>Postal code not found This value will appear if Status.Code=UnrecognizedPostalCode.</p> <p>PerformUSProcessing disabled This value will appear if Status.Code=DisabledCoder.</p> <p>PerformCanadianProcessing disabled This value will appear if Status.Code=DisabledCoder.</p>

Get City State Province Loqate

Get City State Province Loqate returns a city and state/province for a given input postal code.

This stage is part of the Spectrum Universal Adresse.

Input

The following table shows the input fields.

Table 85: Get City State Province Loqate Input

Field Name	Description
Country	<p>The country code or name, in any of the following formats:</p> <ul style="list-style-type: none"> • 2-digit ISO country code • 3-digit UPU Country code • English country name <p>For a list of ISO codes, see ISO Country Codes and Module Support.</p>
PostalCode	The postal code for the address.

Options

Table 86: Get City State Province Loqate Options

Field Name	Description / Valid Values
Database	Specifies the database to be used for address processing. Only databases that have been defined in the Database Resources panel in the Management Console are available.
Maximum records to return	The maximum number of addresses that Get City State Province Loqate should return. The default is 10.

Field Name	Description / Valid Values
Script/Alphabet	<p>Specifies the alphabet or script in which the output should be returned. This option is bi-directional and generally takes place from Native to Latin and Latin to Native.</p> <p>Input Script Do not perform transliteration and provide output in the same script as the input (default).</p> <p>Native Output in the native script for the selected country wherever possible.</p> <p>Latin (English) Use English values.</p>
Data license error handling	<p>Specifies how you want Spectrum Technology Platform to respond when a data license error occurs.</p> <p>Fail the job Fail the entire job if a data license error occurs.</p> <p>Fail the record Fail the record(s) for which the data license error occurs and continue processing.</p>

Output

Get City State Province Loqate returns the matching city and state/province for the input postal code as well as a code to indicate the success or failure of the match attempt. If more than one city/state or city/province matches the input postal code, multiple output records are returned.

Table 87: Get City State Province Loqate Output

Field Name	Description
City	The matched city name.
Country	<p>The country in the format determined by what you selected in Country format:</p> <ul style="list-style-type: none"> • ISO Code • UPU Code • English
PostalCode	The input postal code.

Field Name	Description
ProcessedBy	Indicates which address coder processed the address. LOQATE The Loqate coder processed the address.
StateProvince	The validated state/province or its abbreviated value.
Status	Reports the success or failure of the match attempt. null Success F Failure
Status.Code	The reason for failure, if there is one. The only valid value is: • UnrecognizedPostalCode
Status.Description	The description of the failure. The only valid value is: Postal code not found This value will appear if Status.Code=UnrecognizedPostalCode.

Get Postal Codes

Get Postal Codes allows you to look up the postal codes for a particular city. The service takes a city, state, and country as input and returns the postal codes for that city. The input must be exactly correct in order to return postal codes.

Note: Get Postal Codes only works with U.S. addresses.

Get Postal Codes is part of the Spectrum Universal Address.

Input

Get Postal Codes takes a city, state/province, and country as input.

Table 88: Get Postal Codes Input

Field Name	Description
City	<p>The city whose postal codes you want to look up.</p> <p>You may put the city and state in the City field. If you do this, you must leave the StateProvince field blank.</p> <p>The total length of the City and StateProvince fields cannot exceed 100 characters.</p>
StateProvince	<p>The state or province of the city whose postal codes you want to look up.</p> <p>You may also put the state in the City field instead of the StateProvince field.</p> <p>The total length of the City and StateProvince fields cannot exceed 100 characters.</p>
Country	<p>The country code or name of the city whose postal codes you want to look up. The only valid value is US.</p>

Options

Table 89: Get Postal Codes Options

Option	Description
Database	<p>Specifies the database to be used for postal code look-ups. Only databases that have been defined in the US Database Resources panel in the Management Console are available.</p>
Include non-mailing city	<p>Specifies whether or not to include postal codes for the city's non-mailing city names. A non-mailing city name is an alternate name for the primary city name. For example, Hollywood is a non-mailing city name for Los Angeles.</p>
Include city type	<p>Specifies whether or not to return the city type in the output. If enabled, the city type is returned in the City.Type field.</p>

Output

Get Postal Codes returns the postal codes for a specified city. Each postal code is returned in a separate record along with the data listed in the following table.

Table 90: Get Postal Codes Output

Field Name	Description
City.Type	<p>The USPS® city type (U.S. addresses only). The city type is determined by looking at the ZIP Code and the city name. For example, the city Lanham MD has the postal codes 20703, 20706, and 20784. Lanham is the primary city in 20703 and 20706 but is a vanity city in 20784.</p> <p>This field column is only populated if Include city type is checked. The possible values are:</p> <p>V Vanity (non-mailing) city name.</p> <p>P Primary. The city name is the primary mailing city name.</p> <p>S Secondary. The city name is an alternate city name but is acceptable. A city can have multiple secondary city names.</p>
PostalCode	A postal code in the specified city.
ProcessedBy	Because this service only works for U.S. addresses, ProcessedBy will always contain one value: USA.
Status	<p>Reports the success or failure of the match attempt.</p> <p>null Success</p> <p>F Failure</p>
Status.Code	<p>Reason for failure, if there is one. One of the following:</p> <ul style="list-style-type: none"> CountryNotSupported UnableToLookup

Field Name	Description
Status.Description	<p>Description of failure.</p> <ul style="list-style-type: none"> • Input country is not supported • Input city was blank • Input city & state / province was blank, or no match found • City-state mismatch (different spelling found, or city-state was a vanity name and vanity matching was not allowed, or city-state did not match ZIP Code)

Get Postal Codes Loqate

Get Postal Codes Loqate allows you to look up the postal codes for a particular city. The service takes a city, state, and country as input and returns the postal codes for that city. The input must be exactly correct in order to return postal codes.

Get Postal Codes Loqate is part of Spectrum Universal Address.

Input

Get Postal Codes Loqate takes a city, state/province, and country as input.

Table 91: Get Postal Codes Loqate Input

Field Name	Description / Valid Values
City	<p>The city whose postal codes you want to look up.</p> <p>You may put the city and state in the City field. If you do this, you must leave the StateProvince field blank.</p>
Country	<p>The country code or name, in any of the following formats:</p> <ul style="list-style-type: none"> • 2-digit ISO country code • 3-digit UPU Country code • English country name <p>For a list of ISO codes, see ISO Country Codes and Module Support.</p>
StateProvince	<p>The state or province of the city whose postal codes you want to look up.</p> <p>You may also put the state in the City field instead of the StateProvince field.</p>

Options

Table 92: Get Postal Codes Loqate Options

Option	Description				
Database	Specifies the database to be used for postal code look-ups. Only databases that have been defined in the Management Console are available.				
Data license error handling	Specifies how you want Spectrum Technology Platform to respond when a data license error occurs. <table border="0" style="margin-left: 20px;"> <tr> <td>Fail the job</td> <td>Fail the entire job if a data license error occurs.</td> </tr> <tr> <td>Fail the record</td> <td>Fail the record(s) for which the data license error occurs and continue processing.</td> </tr> </table>	Fail the job	Fail the entire job if a data license error occurs.	Fail the record	Fail the record(s) for which the data license error occurs and continue processing.
Fail the job	Fail the entire job if a data license error occurs.				
Fail the record	Fail the record(s) for which the data license error occurs and continue processing.				

Output

Get Postal Codes Loqate returns the postal codes for a specified city. Each postal code is returned in a separate record along with the data listed in the following table.

Table 93: Get Postal Codes Loqate Output

Field Name	Description / Valid Values				
PostalCode	A postal code in the specified city.				
ProcessedBy	Indicates which address coder processed the address. <table border="0" style="margin-left: 20px;"> <tr> <td>LOQATE</td> <td>The Loqate coder processed the address.</td> </tr> </table>	LOQATE	The Loqate coder processed the address.		
LOQATE	The Loqate coder processed the address.				
Status	Reports the success or failure of the match attempt. <table border="0" style="margin-left: 20px;"> <tr> <td>null</td> <td>Success</td> </tr> <tr> <td>F</td> <td>Failure</td> </tr> </table>	null	Success	F	Failure
null	Success				
F	Failure				

Field Name	Description / Valid Values
Status.Code	Reason for failure, if there is one. One of the following: <ul style="list-style-type: none"> InvalidCountry UnableToLookup
Status.Description	Description of failure. <ul style="list-style-type: none"> Input country is not supported Input city was blank Input city & state / province was blank, or no match found

Validate Address

Validate Address standardizes and validates addresses using postal authority address data. It can correct information and format the address using the format preferred by the applicable postal authority. It also adds missing postal information, such as postal codes, city names, state/province names, and more.

Validate Address also returns result indicators about validation attempts, such as whether or not it validated the address, the level of confidence in the returned address, the reason for failure if the address could not be validated, and more.

During address matching and standardization, Validate Address separates address lines into components and compares them to the contents of the Universal Addressing Module databases. If a match is found, the input address is *standardized* to the database information. If no database match is found, it optionally *formats* the input addresses. The formatting process attempts to structure the address lines according to the conventions of the appropriate postal authority.

Validate Address is part of the Universal Addressing Module.

The ValidateMailingAddressUSCAN API analyzes and compares the input addresses against the known address databases for US and CANADA only, to output standardized details along with DPV and RDI. It also returns parsed address fields and field validation codes. It corrects addresses, adds missing postal information, and formats the address per rules of the applicable postal authority.

Credit Calculation

0.33 credit is deducted for each address.

HTTP Method

POST

Input

Validate Address takes an address as input. All addresses use this format regardless of the address's country. See [Address Line Processing for U.S. Addresses](#) on page 626 for important information about how address line data is processed for U.S. addresses.

Table 94: Input Format

Field Name	Format	Description
AddressLine1	String [50]	The first address line.
AddressLine2	String [50]	The second address line.
City	String [50]	The city name. For U.S. addresses only, you may put the city, state, and ZIP Code™ in the City field. If you do this, you must leave the StateProvince and PostalCode fields blank.
StateProvince	String [50]	The state or province. For U.S. addresses only, you may put the state in the City field instead of the StateProvince field.
PostalCode	String [10]	The postal code for the address in one of the following formats: 99999 99999-9999 A9A9A9 A9A 9A9 9999 999 For U.S. addresses only, you may put the ZIP Code™ in the City field. For U.S. addresses only, if the city/state/ZIP Code™ is in the PostalCode field, Validate Address may parse the data and successfully process the address. For best results, put this data in the appropriate fields (City, StateProvince, and PostalCode).

Field Name	Format	Description
Country	String [50]	The country code or name, in any of the following formats: <ul style="list-style-type: none"> • Two-character ISO 3166-1 Alpha 2 country code • Three-character ISO 3166-1 Alpha 3 country code • English country name • French country name • German country name • Spanish country name
FirmName	String [50]	The company or firm name.
USUrbanName	String [50]	The U.S. address urbanization name. This is used primarily for Puerto Rico addresses.
CanLanguage	String	For Canadian addresses only, indicates whether the address is in English or French, if the option the Determine language using field on the Canadian Address Options tab is set to <code>CanLanguage</code> input field. If this field is blank, the address is formatted in English. If the field contains any non-blank value, the address is formatted in French. Note that addresses in Quebec are always formatted in French regardless of the value in this field.

Address Line Processing for U.S. Addresses

The input fields AddressLine1 through AddressLine4 are handled differently for U.S. addresses depending on whether the firm name extraction or urbanization code extraction options are enabled. If either of these options is enabled, Validate Address will look at the data in all four fields to validate the address and extract the requested data (firm name and/or urbanization code). If neither of these options is enabled, Validate Address uses only the first two non-blank address line fields in its validation attempt. The data in the other address line fields is returned in the output field AdditionalInputData. For example,

AddressLine1: A1 Calle A
AddressLine2:
AddressLine3: URB Alamar
AddressLine4: Precisely

In this address, if either firm name extraction or urbanization code extraction were enabled, Validate Address would examine all four address lines. If neither firm name extraction nor urbanization code extraction were enabled, Validate Address would examine AddressLine1 and AddressLine3 (the first

two non-blank address lines) and attempt to validate the address using that data; the data in AddressLine4 would be returned in the output field AdditionalInputData.

Options

Output Data Options

The following table lists the options that control the type of information returned by Validate Address. Some of these options can be overridden for Canadian addresses. For more information, see [Canadian Address Options](#) on page 648.

Table 95: Output Data Options

Option	Description
Include a standard address	<p>Returns 1 to 4 lines of address data plus city, state, postal code, firm name, and urbanization name information. Each address line represents an actual line of the address as it would appear on an envelope. For more information, see Standard Address Output on page 658.</p> <p>If Validate Address could validate the address, the address lines contain the standardized address. When addresses are standardized, punctuation is removed, directionals are abbreviated, street suffixes are abbreviated, and address elements are corrected.</p> <p>If Validate Address could not validate the address, the address lines contain the address as it appeared in the input ("pass through" data). Non-validated addresses are always included as pass through data in the address line fields even if you uncheck this option.</p>
Include matched address elements	<p>Each part of the address, such as house number, street name, street suffix, directionals, and so on is returned in a separate field. For more information, see Parsed Address Elements Output on page 729. Note that if you select this option and also select Return normalized data when no match is found, the address elements will contain the input address for addresses that could not be validated.</p>
Include postal information	<p>Output addresses contain various additional data for each validated address. For more information, see Postal Data Output on page 663.</p>

Option	Description
Include standardized input address elements	<p>This option returns the input address in parsed form regardless of whether or not Validate Address is able to validate the address. Each part of the input address, such as house number, street name, street suffix, directionals, and so on is returned in a separate field.</p> <p>Selecting this option differs from selecting the combination of Include matched address elements/Return normalized data when no match is found in that Return standardized input address elements returns all input address in parsed form, not just input that could not be validated. For more information, see Parsed Input on page 732.</p>
Return normalized data when no match is found	<p>Specifies whether to return a formatted address when an address cannot be validated. The address is formatted using the preferred address format for the address's country. If this option is not selected, the output address fields are blank when the address cannot be validated.</p> <p>Note: This option applies only to U.S. and Canadian addresses. Formatted data will not be returned for any other address.</p> <p>Formatted addresses are returned using the format specified by the Include a standard address, Include address line elements, and Include postal information check boxes. Note that if you select Include address line elements, the parsed address elements will contain the parsed, validated address for addresses that could be validated. If the address could not be validated the parsed address elements will contain the input address in parsed form. If you always want the output to contain the input address in parsed form, regardless of whether or not Validate Address could validate the address, select Include standardized input address elements.</p> <p>If you check this option, you must select Include a standard address and/or Include address line elements.</p>

Option	Description
Return street name alias	<p>For U.S. addresses only, specifies how to handle street name aliases used in the input. A street alias is an alternate name for a street and applies only to a specific range of addresses on the street.</p> <p>If you enable this option, street name aliases used in the input will appear in the output. If you do not enable this option, street name aliases in the input will be converted to the base street name in the output, with the following exceptions:</p> <ul style="list-style-type: none"> • If a preferred alias is used in input the preferred alias will always be used in output. • Changed aliases used in input are always converted to the base street name in output. <p>This is one of three options that control how Validate Address handles street name aliases. The other two are Preferred street name alias processing and Abbreviated street name alias processing.</p> <p>Note: If Abbreviated street name alias processing is enabled, the abbreviated alias will always appear in the output even if you have Return street name alias disabled.</p>
Return address data blocks	<p>Specifies whether to return a formatted version of the address as it would be printed on a physical mailpiece. Each line of the address is returned in a separate address block field. There can be up to nine address block output fields: AddressBlock1 through AddressBlock9.</p> <p>For example, this input address:</p> <p>AddressLine1: 4200 Parliament Place AddressLine2: Suite 600 City: Lanham StateProvince: MD PostalCode: 20706</p> <p>Results in this address block output:</p> <p>AddressBlock1: 4200 PARLIAMENT PL STE 600 AddressBlock2: LANHAM MD 20706-1882 AddressBlock3: UNITED STATES OF AMERICA</p> <p>Validate Address formats the address into address blocks using postal authority standards. The country name is returned using the Universal Postal Union country name. Note that the option Country format does not affect the country name in the address block, it only affects the name returned in the Country output field.</p> <p>For addresses outside the U.S. and Canada, if Validate Address is unable to validate the address, no address blocks are returned. For addresses in the U.S. and Canada, address blocks are returned even if validation fails.</p>

Obtaining Congressional Districts

Validate Address can determine the U.S. congressional district for an address.

To obtain congressional districts, select the **Include postal information** check box on the **Output Data Options** tab. This will return a variety of data about the address, including the congressional district. For information on the specific data that this option will return, see [Postal Data Output](#) on page 663.

Table 96: Congressional District Output

Field Name	Description
USCongressionalDistrict	Congressional district number. If the address is a non-state address (for example Puerto Rico or Washington D.C.) this field is blank.

Obtaining County Names

Validate Address can determine the county where a particular address is located and return the county name.

Note: County names are available for U.S. addresses only.

To obtain county names, select the **Include postal information** check box on the **Output Data Option** tab. This will return a variety of data about the address, including county names. For information on the specific data that this option will return, see [Postal Data Output](#) on page 663.

Table 97: County Name Output

Field Name	Description
USCountyName	County name

Obtaining FIPS County Numbers

Federal Information Processing Standards (FIPS) county numbers are numbers that identify each county in a state. Note that these numbers are only unique at the state level, not the national level. For more information, see <http://www.census.gov>.

Note: FIPS county numbers are available for U.S. addresses only.

To obtain FIPS county numbers, select the **Include postal information** check box on the **Output Data Options** tab. This will return a variety of data about the address, including FIPS county numbers. For information on the specific data that this option will return, see [Postal Data Output](#) on page 663.

Table 98: FIPS County Number Output

Field Name	Description
USFIPSCountyNumber	FIPS (Federal Information Processing Standards) county number

Obtaining Carrier Route Codes

Carrier route codes are unique identifiers assigned to each mail carrier who delivers mail, allowing unique identification of each U.S. delivery route. Validate Address can return the code that represents an addressee's carrier route.

Note: Carrier route codes are available for U.S. addresses only.

To obtain carrier route codes, select the **Include postal information** check box on the **Output Data Options** tab. This will return a variety of data about the address, including carrier route codes. For information on the specific data that this option will return, see [Postal Data Output](#) on page 663.

Table 99: Carrier Route Code Output

Field Name	Description
USCarrierRouteCode	Carrier route code

Creating Delivery Point Barcodes

A Delivery Point Barcode (DPBC) is a POSTNET™ barcode representation of the address. It consists of 62 bars with beginning and ending frame bars and five bars each for the ZIP + 4® Code, a value calculated based on the street address number, and a correction digit. The DPBC allows automated sortation of letter mail to the carrier level in walk sequence. Validate Address generates the data you need to assemble a DPBC.

Note: Delivery Point Barcodes are available for U.S. addresses only. For more information on Delivery Point Barcodes, see <http://www.usps.com>.

To generate the data needed to assemble a DPBC, select the **Include postal information** check box on the **Output Data Options** tab. This will return a variety of data about the address, including data needed to construct DPBCs. For information on the specific data that this option will return, see [Postal Data Output](#) on page 663.

Table 100: Delivery Point Barcode Output

Field Name	Description
PostalBarCode	The delivery point portion of the delivery point barcode.
USBCCheckDigit	Check-digit portion of the 11-digit delivery point barcode.

To assemble a DPBC you concatenate the values found in the Validate Address output fields as follows:

PostalCode.Base + PostalCode.Addon + PostalBarcode + USBCCheckDigit

For example, if you have the following:

- **PostalCode.Base** = 49423
- **PostalCode.Addon** = 4506
- **PostalBarcode** = 29
- **USBCCheckDigit** = 2

The assembled barcode would be:

494234506292

Default Options

The following table lists the options that control the format and processing of addresses. These are called "default options" because by default they apply to all addresses. Some of these options can be overridden for Canadian addresses. For more information, see [Canadian Address Options](#) on page 648.

Table 101: Default Options

Option	Description
Casing	<p>Specifies the casing of the output address. One of the following:</p> <p>Mixed Returns the output in mixed case (default). For example:</p> <p style="padding-left: 40px;">123 Main St Mytown FL 12345</p> <p>Upper Returns the output in upper case. For example:</p> <p style="padding-left: 40px;">123 MAIN ST MYTOWN FL 12345</p>
Insert postal code separation character	<p>Specifies whether to use separators (spaces or hyphens) in ZIP™ Codes or Canadian postal codes.</p> <p>For example, a ZIP + 4® Code with the separator would be 20706-1844 and without the separator it would be 207061844. A Canadian postal code with the separator would be P5E"1S7 and without the separator it would be P5E1S7.</p> <p>Note: Spaces are used in Canadian postal codes and hyphens in U.S. ZIP + 4® Codes.</p>
Output multinational characters	<p>Specifies whether or not to return multinational characters, including diacritical marks such as umlauts or accents. (Not supported for U.S. addresses).</p>

Option	Description
Secondary address placement	<p data-bbox="675 380 1425 527">Specifies where to place secondary address information for U.S. addresses. Secondary address information refers to apartment numbers, suite numbers, and similar designators. For example, in this address the secondary address information is "Apt 10E" and the primary address information is "424 Washington Blvd".</p> <p data-bbox="675 548 902 638">Apt 10E 424 Washington Blvd Springfield MI 49423</p> <p data-bbox="683 659 1341 722">Same line as address Place both primary and secondary address information in AddressLine1 (default).</p> <p data-bbox="683 743 1328 827">Separate address line Place the primary address information in AddressLine1 and the secondary address information in AddressLine2.</p> <p data-bbox="683 848 1417 1073">Dual address separation Place both primary and secondary address information in AddressLine1 and place dropped information from dual addresses in AddressLine2. A dual address is an address that contains both street information and PO Box/Rural Route/Highway Contract information. For more information, see About Dual Address Logic on page 636.</p>

Option	Description
City format	<p>Specifies how to format city names that have short city name or non-mailing city name alternatives. Applies to U.S. and Canadian addresses.</p> <p>Short Returns the USPS®-approved abbreviation for the city, if there is one. The USPS® provides abbreviations for city names that are 14 characters long or longer. City abbreviations are 13 characters or less and can be used when there is limited space on the mailing label. If there is no short city name for the city, then the full city name is returned.</p> <p>Long Returns the long city name (default).</p> <p>Standard Returns the abbreviated city name only if an abbreviated city name is used in the input address. If the input address does not use a short city name, either the long or short city name could be returned, depending on USPS® regulations for the particular city. Select this option if you are performing a CASS™ test.</p> <p>Non-Mailing (Vanity) Output the non-mailing city name (the vanity name) if the input city name is a non-mailing city name. For example, "Hollywood" is a non-mailing city name for "Los Angeles". If you do not select this option and the input city name is a non-mailing city name the long version of the mailing city is returned.</p>
Country format	<p>Specifies the format to use for the country name returned in the Country output field. For example, if you select English, the country name "Deutschland" would be returned as "Germany".</p> <p>English Names Use English country names (default).</p> <p>Spanish Names Use Spanish country names.</p> <p>French Names Use French country names.</p> <p>German Names Use German country names.</p> <p>ISO Codes Use two-letter ISO abbreviation for the countries instead of country names.</p> <p>UPU Codes Use Universal Postal Union abbreviation for the countries instead of country names.</p>

Option	Description
Default country	Specifies the default country. You should specify the country where most of your addresses reside. For example, if most of the addresses you process are in Canada, specify Canada. Validate Address uses the country you specify to attempt validation when it cannot determine the country from the StateProvince, PostalCode, and Country address fields.
Dual address logic	<p>Indicates how to return a match if multiple non-blank address lines are present or multiple address types are on the same address line. (U.S. addresses only.)</p> <p>Normal Match (Default) USPS® CASS™ regulations determine the address returned based on the following order of priority:</p> <ol style="list-style-type: none"> 1. PO Box 2. Firm 3. Highrise 4. Street 5. Rural Route 6. General Delivery <p>Street Match Return a street match, regardless of the address line.</p> <p>PO Box Match Return a PO Box match, regardless of the address line.</p> <p>For more information, see About Dual Address Logic on page 636.</p>

About Dual Address Logic

For U.S. addresses only, the **Dual address logic** option controls whether Validate Address should return a street match or a PO Box/Rural Route/Highway Contract match when the address contains both street and PO Box/Rural Route/Highway Contract information in the same address line.

Note: The **Dual address logic** option has no effect if the street information is in a different address line input field than the PO Box/Rural Route/Highway Contract information.

For example, given the following input address:

AddressLine1: 401 N Main St Apt 1 POB 1
 City: Kemp
 StateProvince: TX
 PostalCode: 75143

Validate Address would return one of the following:

- If **Dual address logic** is set to either `Normal Match` or `PO Box Match`:

AddressLine1: PO Box 1
 City: Kemp
 StateProvince: TX
 PostalCode: 75143-0001

- If **Dual address logic** is set to `Street Match`:

AddressLine1: 401 N Main St Apt 1
 City: Kemp
 StateProvince: TX
 PostalCode: 75143-4806

The address data that is not used to standardize the address can be returned in one of two places:

- **AddressLine2**—The address information not used to standardize the address is returned in the **AddressLine2** field if you select `Dual address separation` in the **Secondary address placement** field. For more information, see [Default Options](#) on page 632. For example, if you choose to return a street match for dual addresses,

AddressLine1: 401 N Main St Apt 1
 AddressLine2: PO Box 1
 City: Kemp
 StateProvince: TX
 PostalCode: 75143-0001

- **AdditionalInputData**—If you do not select `Dual address separation` in the **Secondary address placement** field then the address information not used to standardize the address is returned in the **AdditionalInputData** field. For more information on this option, see [Default Options](#) on page 632. For example, if you choose to return a street match for dual addresses,

AddressLine1: 401 N Main St Apt 1
 City: Kemp
 StateProvince: TX
 PostalCode: 75143-0001
 AdditionalInputData: PO Box 1

Address information that is dropped can be retrieved by setting **Secondary address placement** to `Dual address separation`. For more information, see [Default Options](#) on page 632 .

Returning Multiple Matches

If `Validate Address` finds multiple address in the postal database that are possible matches for the input address, you can have `Validate Address` return the possible matches. For example, the following address matches multiple addresses in the U.S. postal database:

PO BOX 1
New York, NY

Options

To return multiple matches, use the options described in the following table.

Table 102: Multiple Match Option

Option Name	Description
Return multiple addresses	Indicates whether or not to return multiple address for those input addresses that have more than one possible match.
Maximum results	<p>Next to the Return multiple addresses check box, enter a number between 1 and 10 that indicates the maximum number of addresses to return.</p> <p>The default value is 1.</p> <p>Note: The difference between unchecking Return multiple addresses and checking Return multiple addresses and specifying a maximum number of results of 1 is that a multiple match will return a failure if Return multiple addresses is unchecked, whereas a multiple match will return one record if Return multiple addresses is checked and the maximum number of results is 1.</p>
Include result codes for individual fields	To identify which output addresses are candidate addresses, you must check Include result codes for individual fields on the Output Data tab. When you do this, records that are candidate addresses will have one or more "M" values in the field-level result indicators.

Output

When you choose to return multiple matches, the addresses are returned in the address format you specify. For information on specifying address format, see [Output Data Options](#) on page 627. To identify which records are the candidate addresses, look for multiple "M" values in the field-level result indicators. For more information, see [Field-Level Result Indicators](#) on page 670.

U.S. Address Options

Option Name	Description
Enable U.S. address processing	<p>Specifies whether to process U.S. addresses. If you enable U.S. address processing Validate Address will attempt to validate U.S. addresses. If you disable U.S. address processing, U.S. addresses will fail, meaning they are returned with an "F" in the Status output field. The output field Status.Code will say "DisabledCoder." If you are not licensed for U.S. address processing you must disable U.S. address processing in order for your jobs to complete successfully, regardless of whether or not they contain U.S. addresses.</p> <p>Note: You must have a valid license for U.S. address processing to successfully process U.S. addresses. If you enable U.S. address processing but are not licensed for this feature, or your license has expired, your entire job will fail.</p>
Database	<p>Specifies which database to use for validating U.S. addresses. Only databases that have been defined in the US Database Resources panel in the Management Console are available.</p>
Line of travel	<p>Enhanced Line of Travel (eLOT) processing assigns a Line of Travel sequence code to your addresses. Note that addresses are not sorted into eLOT sequence but they are assigned a Line of Travel sequence code that allows you to sort addresses into eLOT sequence.</p> <p>To perform eLOT processing you must have the eLOT database installed.</p> <p>For a listing of the output fields returned by this option, see Enhanced Line of Travel Output on page 684.</p>
Residential Delivery Indicator processing	<p>Residential Delivery Indicator (RDI™) processing checks if an address is a residential address (not a business address). To perform RDI™ processing, you must have the RDI™ database installed.</p> <p>If you enable both DPV® and RDI™ processing, RDI™ information is only returned if the address is a valid delivery point. If DPV® does not validate the address no RDI™ data is returned.</p>
Enhanced street matching	<p>Enhanced Street Matching (ESM) applies additional matching logic to correct misspelled or complex street names and obtain a match. ESM enables more addresses to be validated but it reduces performance. You cannot perform ESM when ASM is enabled.</p>

Option Name	Description
All street matching	<p>All Street Matching (ASM) applies ESM processing as well as additional matching logic to correct errors in street names and obtain a match. It is effective at matching streets when the first letter of the street is incorrect. ASM provides the best address validation but reduces performance.</p>
Delivery Point Validation & CMRA	<p>Delivery Point Validation (DPV[®]) validates that a specific address exists, as opposed to validating that a specific address is within a range of valid addresses. CMRA processing checks if an address is for a mailbox rented from a private company, referred to as a Commercial Mail Receiving Agent (CMRA).</p> <p>To perform DPV and CMRA processing, you must have the DPV database installed. The DPV database contains both DPV and CMRA data.</p> <p>For a listing of the output fields returned by this option, see DPV and CMRA Output on page 687.</p>
LACS/Link conversion	<p>The USPS[®] Locatable Address Conversion System (LACS) allows you to correct addresses that have changed as a result of a rural route address converting to street-style address, a PO Box renumbering, or a street-style address changing. When enabled, LACS^{Link} processing is attempted for addresses that could not be validated, or addresses were validated and flagged for LACS^{Link} conversion.</p> <p>To perform LACS^{Link} processing, you must have the LACS^{Link} database installed.</p> <p>For a listing of the output fields returned by this option, see LACSLink Output on page 685</p>
Early Warning System	<p>The Early Warning System (EWS) uses the USPS[®] EWS File to validate addresses that are not in the ZIP + 4[®] database.</p> <p>To perform EWS processing, you must have the EWS database installed.</p> <p>If an input address matches an address in the EWS file, the following record-level result indicators are returned:</p> <ul style="list-style-type: none"> • Status="F" • Status.Code="EWSFailure" • Status.Description="Address found in EWS table"

Option Name	Description
-------------	-------------

Firm name extraction	
----------------------	--

Option Name

Description

Specifies whether to extract the firm name from AddressLine1 through AddressLine4 and place it in the FirmName output field. This option works in cases where the input record's FirmName field is blank and there is more than one address line.

To identify firm names in address lines, the address lines are scanned for keywords and patterns that identify which fields are address lines and which are FirmName lines. Since this is done based on patterns, fields may be misidentified. The following tips can help ensure optimal firm extraction:

- If possible, place the primary address elements in AddressLine1, the secondary elements in AddressLine2, Urbanization in AddressLine3, and firm in AddressLine4. If the address has no urbanization code, then place the firm name in AddressLine3 and leave AddressLine4 blank. For example,

AddressLine1: 4200 Parliament Place

AddressLine2: Suite 600

AddressLine3: Precisely

AddressLine4: <blank>

- When you define just two address lines, AddressLine2 is assigned to the secondary address most of the time. If you want to increase the chance that AddressLine2 will be treated as a firm name, put the firm name in AddressLine3 and leave AddressLine2 blank.
- Numbers in a firm name (such as the "1" in "1 Stop Software") will increase the likelihood that the field will be treated as an address line.

Here are some examples of firm name extraction:

- In this example, AddressLine2 would get extracted into the FirmName output field

FirmName: <blank>

AddressLine1: 4200 Parliament Place Suite 600

AddressLine2: International Goose Feathers inc.

- In this example, AddressLine3 would get extracted into the FirmName output field.

FirmName: <blank>

AddressLine1: 4200 Parliament Place

AddressLine2: Suite 600

AddressLine3: Precisely

- In this example, AddressLine3 would be placed in the AdditionalInputData output field. The firm name would not be extracted because the FirmName input field is not blank.

FirmName: International Goose Feathers Inc.

AddressLine1: 4200 Parliament Place

AddressLine2: Suite 600

AddressLine3: Precisely

- In this example, no firm name would be extracted because there is only one non-blank address line, which is always treated as the primary address element.

FirmName: <blank>

AddressLine1: 4200 Parliament Place Suite 600

Option Name	Description
	<ul style="list-style-type: none"> In this example, AddressLine2 would be treated as a secondary address element because the numeral "1" causes that field to be treated as a secondary address element. <p>FirmName: <blank> AddressLine1: 4200 Parliament Place Suite 600 AddressLine2: 1 Stop Software</p>
U.S. urbanization name extraction	<p>Specifies whether to extract the urbanization name from AddressLine1 through AddressLine4 and place it in the USUrbanName output field. This option works in cases where the input record's USUrbanName field is blank and there is more than one address line.</p> <p>To identify urbanization names, the address lines are scanned for keywords and patterns that identify which fields are address lines and which are urbanization name lines. Since this is done based on patterns, it is possible for fields to be incorrectly identified. To help ensure optimal urbanization extraction, place the primary address elements in AddressLine1, the secondary elements in AddressLine2, Urbanization in AddressLine3, and firm in AddressLine4, if possible. For example,</p> <p>AddressLine1: A1 Calle A AddressLine2: AddressLine3: URB Alamar AddressLine4: Precisely</p>
Suite/Link support	<p>Specifies whether to perform Suite^{Link™} processing.</p> <p>Suite^{Link} corrects secondary address information for U.S. business addresses whose secondary address information could not be validated. If Suite^{Link} processing is enabled, the firm name is matched to a database of known firm names and their secondary address information.</p> <p>For example,</p> <p>Firm Name: Precisely Address Line 1: 4200 Parliament Place Address Line 2: STE 1 Postal Code: 20706</p> <p>In this case, Suite^{Link} processing would provide the correct suite number:</p> <p>Firm Name: Precisely Address Line 1: 4200 Parliament Pl Address Line 2: STE 500 Postal Code: 20706-1844</p> <p>To perform Suite^{Link™} processing, you must have the Suite^{Link™} database installed.</p> <p>For a listing of fields returned by this option, see SuiteLink Output on page 689.</p>

Option Name	Description
Preferred alias street name processing	<p data-bbox="548 380 1214 407">Specifies whether to use a street's preferred alias in the output.</p> <p data-bbox="548 426 1425 483">Street name aliases in the United States are alternative names given to sections of a street. There are four types of street name aliases:</p> <ul data-bbox="548 501 1425 835" style="list-style-type: none"> <li data-bbox="548 501 1425 558">• Preferred—A preferred alias is the street name preferred locally. It typically applies only to a specific range of addresses on the street. <li data-bbox="548 569 1425 680">• Abbreviated—An abbreviated alias is a variation of the street name that can be used in cases where the length of AddressLine1 is longer than 31 characters. For example, the street name 1234 BERKSHIRE VALLEY RD APT 312A could be abbreviated to 1234 BERKSHIRE VLLY RD APT 312A. <li data-bbox="548 690 1425 772">• Changed—There has been an official street name change and the alias reflects the new name. For example if SHINGLE BROOK RD is changed to CANNING DR, then CANNING DR would be a changed alias type. <li data-bbox="548 783 1425 835">• Other—The street alias is made up of other names for the street or common abbreviations of the street. <p data-bbox="548 863 1305 890">The non-alias version of the street name is called the base street name.</p> <p data-bbox="548 909 1398 966">If the preferred alias is used in the input then the preferred alias will be the street name in the output regardless of whether you enable this option.</p> <p data-bbox="548 984 1425 1066">This is one of three options that control how Validate Address handles street name aliases. The other two are Return street name alias and Abbreviated street name alias processing.</p> <p data-bbox="548 1085 1425 1230">In most cases, if you select both Preferred street name alias processing and Abbreviated street name alias processing, and Validate Address finds both a preferred and an abbreviated alias in the postal database, the abbreviated alias will be used in the output. The exception to this rule is if the input street name is a preferred alias. In this case, the preferred alias will be used in the output.</p> <p data-bbox="631 1249 1425 1331">Note: If the input address contains a street name alias of type "changed" the output address will always contain the base street name regardless of the options you specify.</p>
Abbreviated alias street name processing	<p data-bbox="548 1430 1425 1486">Specifies whether to use a street's abbreviated alias in the output if the output address line is longer than 31 characters.</p> <p data-bbox="548 1505 1425 1587">This is one of three options that control how Validate Address handles street name aliases. The other two are Return street name alias and Preferred street name alias processing.</p> <p data-bbox="631 1606 1425 1688">Note: If a preferred alias is specified in the input, the output street name will always be the preferred alias, even if you enable abbreviated street name alias processing.</p> <p data-bbox="631 1724 1425 1808">Note: If the input address contains a street name alias of type "changed" the output address will always contain the base street name regardless of the options you specify.</p>

Option Name	Description								
Determine if delivery point is active	<p>Determines the "no stat" status of an address. An address is considered "no stat" if it exists but cannot receive mail, and therefore is not counted as a delivery statistic on a carrier's route (hence the term "no stat"). Examples include buildings under construction or those that the letter carrier has identified as not likely to receive mail.</p> <p>Note: You must enable DPV processing to use this option.</p> <p>The result is returned in the DPVNoStat field. For more information see LACSLink Output on page 685</p>								
Determine if address is vacant	<p>Determines if the location has been unoccupied for at least 90 days.</p> <p>Note: You must enable DPV processing to use this option.</p> <p>The result is returned in the DPVVacant field. For more information see LACSLink Output on page 685</p>								
Suppress zip+4 carrier route R777	<p>Specifies whether to suppress addresses with Carrier Route R777. These addresses are phantom routes and are not eligible for street delivery. Since these addresses are assigned a ZIP + 4[®] code by the USPS[®], Validate Address marks these addresses as deliverable. Select this option if you do not want addresses with Carrier Route R777 marked as deliverable. This will cause the following actions:</p> <ul style="list-style-type: none"> • No ZIP + 4 code is assigned • Address is not counted on the USPS Form 3553 (CASS Summary Report) • DPV Footnote of R7 is returned 								
Street matching	<p>Specifies the algorithm to use when determining if an input address matches an address in the postal database. One of the following:</p> <table border="0"> <tr> <td data-bbox="558 1402 623 1428">Exact</td> <td data-bbox="808 1402 1398 1428">The input street name must match the database exactly.</td> </tr> <tr> <td data-bbox="558 1449 618 1474">Tight</td> <td data-bbox="808 1449 1154 1474">The matching algorithm is "tight."</td> </tr> <tr> <td data-bbox="558 1495 646 1520">Medium</td> <td data-bbox="808 1495 1289 1520">The matching algorithm is "medium" (default).</td> </tr> <tr> <td data-bbox="558 1541 630 1566">Loose</td> <td data-bbox="808 1541 1166 1566">The matching algorithm is "loose."</td> </tr> </table>	Exact	The input street name must match the database exactly.	Tight	The matching algorithm is "tight."	Medium	The matching algorithm is "medium" (default).	Loose	The matching algorithm is "loose."
Exact	The input street name must match the database exactly.								
Tight	The matching algorithm is "tight."								
Medium	The matching algorithm is "medium" (default).								
Loose	The matching algorithm is "loose."								

Option Name	Description
Firm matching	<p>Specifies the algorithm to use when determining if an input address matches an address in the postal database. One of the following:</p> <p>Exact The input firm name must match the database exactly.</p> <p>Tight The matching algorithm is "tight."</p> <p>Medium The matching algorithm is "medium" (default).</p> <p>Loose The matching algorithm is "loose."</p>
Directional matching	<p>Specifies the algorithm to use when determining if an input address matches an address in the postal database. One of the following:</p> <p>Exact The input directionals, such as the "N" in 123 N Main St., must match the database exactly.</p> <p>Tight The matching algorithm is "tight."</p> <p>Medium The matching algorithm is "medium". Default.</p> <p>Loose The matching algorithm is "loose."</p>
DPV Success Condition	<p>Select the match condition where a DPV result does NOT cause a record to fail.</p> <p>Note: You must enable DPV processing to use this option.</p>
Fail on CMRA match	<p>Treat Commercial Mail Receiving Agency (CMRA) matches as failures?</p> <p>Note: You must enable DPV processing to use this option.</p>
Place PMB elements in	<p>Specifies where Private Mailbox (PMB) information is placed.</p> <p>No AddressLine Do not include the PMB information in Standard Address output (default).</p> <p>AddressLine1 Place the PMB information in AddressLine1. If you choose AddressLine1, you must set the Address Format field to either <i>Combined Unit</i> or <i>Separate Dual Address</i>.</p> <p>AddressLine2 Place the PMB information in AddressLine2. You may not select this option if Generate 3553 Form is checked.</p>

Option Name	Description
Preferred City	Specifies whether the preferred last line city name should be stored.
ZIP+4 Last Line	<p>Store the Preferred Last Line City Name from the USPS ZIP+4 File (Override City Name).</p> <p>Note: If you select this option, Validate Address generates a CASS-certified configuration and the USPS 3553 Report.</p>
USPS City/State	<p>Store the USPS-preferred City Name from USPS City/State File.</p> <p>Note: If you select this option, Validate Address does not generate a CASS-certified configuration and does not generate the USPS 3553 Report.</p>
Primary	<p>Store the Primary City Name from the USPS City/State File.</p> <p>Note: If you select this option, Validate Address does not generate a CASS-certified configuration and does not generate the USPS 3553 Report.</p>

CASS Certified Processing

Validate Address can operate in a CASS Certified™ mode when a specific combination of options are enabled. CASS Certified™ processing enables you to qualify for USPS® postal discounts.

When you use CASS Certified™ processing, Validate Address generates USPS CASS Form 3553. This form must be given to the USPS along with the mailing to qualify for certain discounts. The form contains information about the software you used for CASS processing, information about your name-and-address list, information about your output file, information about the mailer, and other statistics about your mailing. For detailed information about USPS Form 3553, see www.usps.com.

CASS Certified™ processing also generates the USPS CASS Detailed Report, which contains some of the same information as the 3553 report but provides much greater detail about DPV, LACS, and SuiteLink statistics. The USPS CASS Detailed Report is not required for postal discounts and does not need to be submitted with your mailing.

Note: USPS CASS Form 3553 and the USPS CASS Detailed Report are available for batch processing only.

To run Validate Address in CASS Certified™ mode, follow these steps:

1. Validate Address must be in CASS Certified™ mode. If **(Not CASS Certified)** appears at the top of the window, click the **Enable CASS** button. The **Enforce CASS rules** check box will appear.

2. Click **Configure CASS 3553**. The **CASS Report Fields** dialog box appears.
3. Type the **List Processor** company name, **List Name or ID#**, and the **Number of Lists** being processed for this job.
4. Type the **Mailer Name, Address, and City, State, ZIP**.
5. Click **OK**.

The List information will appear in Section B and the Mailer information in Section D of the generated USPS® CASS Form 3553.

6. In Enterprise Designer, drag **CASS3553** from the Reports pallet to the canvas.
7. Double-click the **CASS3553** icon on the canvas.
8. On the **Stages** tab, check the **Validate Address** check box. Note that if you have renamed the Validate Address stage to something else, you should check the box with the name you have given the address validation stage.
9. On the **Parameters** tab, select the format for the report. You can create the report in PDF, HTML, or plain text format.
10. Click **OK**.
11. Repeat steps 6-10 for **CASSDetail** if you want to produce the CASS Detail Report.

Note: You do not need to draw a connector between the Validate Address stage and the reports.

Canadian Address Options

Option Name	Description
Enable Canadian address processing	<p>Specifies whether to process Canadian addresses. If you enable Canadian address processing Validate Address will attempt to validate Canadian addresses. If you disable Canadian address processing, Canadian addresses will fail, meaning they is returned with an "F" in the Status output field. The output field Status.Code will say "DisabledCoder." If you are not licensed for Canadian address processing you must disable Canadian address processing in order for your jobs to complete successfully, regardless of whether or not they contain Canadian addresses.</p> <p>Note: You must have a valid license for Canadian address processing to successfully process Canadian addresses. If you enable Canadian address processing but are not licensed for this feature, or your license has expired, your entire job will fail.</p>

Option Name	Description
Database	<p>Specifies which database you want to use for validating Canadian addresses. To specify a database for Canadian address validation, select a database in the Database drop-down list. Only databases that have been defined in the CAN Database Resources panel in the Management Console are available.</p>
Determine language using	<p>Specifies how to determine the language (English or French) to use to format the address and directional. The following example shows an address formatted in English and French:</p> <p>English: 123 Main St W French: 123 Rue Main O</p> <p>The parameter controls the formatting of the address. It also affects the spelling of the directional but not spelling of the suffix.</p> <p>Street suffix Use the street suffix returned by the matching process to determine the language. The street suffix returned by the matching process, which is used internally by Validate Address during processing, may be different from that in the input address. Ambiguous records are formatted like the input. Default. All addresses in Quebec are formatted using French.</p> <p>CPC database Use the Canadian database to determine the language. The Canadian database contains data from the Canada Post Corporation (CPC). All addresses in Quebec are formatted using French.</p> <p>CanLanguage field Use the CanLanguage input field to determine the language. If there is a non-blank value in this field the address are formatted using French.</p>

Option Name	Description
Default English apt label	<p>For English addresses, specifies the default apartment label to use in the output if there is no apartment label in the input address. This setting is ignored if you select Number in front in the Secondary address format field.</p> <p>Apt Use "Apt" as the label. Default.</p> <p>Apartment Use "Apartment" as the label.</p> <p>Suite Use "Suite" as the label.</p> <p>Unit Use "Unit" as the label.</p>
Default French apt label	<p>For French addresses, specifies the default apartment label to use in the output if there is no apartment label in the input address. This setting is ignored if you select Number in front in the Secondary address format field.</p> <p>App Use "App" as the label. Default.</p> <p>Appartement Use "Appartement" as the label.</p> <p>Bureau Use "Bureau" as the label.</p> <p>Suite Use "Suite" as the label.</p> <p>Unite Use "Unite" as the label.</p>
Prefer house number on postal code conflict	<p>In cases where the house number and postal code are both valid but in conflict, you can force the postal code to be corrected based on the house number by selecting Prefer house number on postal code conflict. If you do not select this option the house number is changed to match the postal code.</p>
Return city alias	<p>Specifies whether or not to return the city alias when the alias is in the input address. This option is disabled when you select Use default option in the City format field.</p>
Abbreviate non-civic keywords	<p>Specifies whether or not non-civic keywords are abbreviated in the output. For example, Post Office Box vs. PO Box.</p>

Option Name	Description
Secondary address format	<p data-bbox="805 380 1422 491">Specifies where to place secondary address information in the output address. Secondary address information refers to apartment numbers, suite numbers, and similar designators.</p> <p data-bbox="805 516 1422 636">Use default option Place apartment information in the location specified in the Secondary address format field in the Default Options tab Default.</p> <p data-bbox="805 661 1422 716">End of address line Place apartment information at the at the end of the AddressLine1 field.</p> <p data-bbox="805 741 1422 858">Front, number only Place the apartment number only (no label) at the beginning of the AddressLine1 field. For example, 400-123 Rue Main</p> <p data-bbox="805 884 1422 1001">Front, number and label Place the apartment number and label at the beginning of the AddressLine1 field. For example, Apt 400 123 Rue Main</p> <p data-bbox="805 1026 1422 1079">Same as input Place apartment information in the same location as the input address.</p>
City format	<p data-bbox="805 1169 1422 1224">Specifies whether to use the long, medium, or short version of the city if the city has a long name. For example,</p> <p data-bbox="805 1249 1154 1331">Long: BUFFALO HEAD PRAIRIE Medium: BUFFALO-HEAD-PR Short: BUFFALO-HD-PR</p> <p data-bbox="805 1356 1422 1579">Use default option Use the default option specified on the Default Options tab, City format field. Default. If you select <i>Non-mailing (vanity)</i> in the City format field, the city is formatted as if you select <i>Long</i> for this option (see below) and check the Return city alias box.</p> <p data-bbox="805 1604 1224 1631">Short Output short city name.</p> <p data-bbox="805 1656 1256 1684">Long Output the long city name.</p> <p data-bbox="805 1709 1295 1736">Medium Output the medium city name.</p> <p data-bbox="805 1761 1422 1801">Same as input Use the same city format as used in the input address. Output is L, M, or S.</p>

Option Name	Description
Place rural route info in	<p data-bbox="805 380 1425 464">Specifies where to place rural route delivery information. An example of an address with rural route delivery information is:</p> <p data-bbox="805 489 1024 548">36 GRANT RD RR 3 ANTIGONISH NS</p> <p data-bbox="805 573 1425 600">In this address, "RR 3" is the rural route delivery information.</p> <p data-bbox="805 615 1425 705">AddressLine1 Place rural route delivery information on the same line as the address, after the address information. Default. For example,</p> <p data-bbox="971 730 1195 758">36 GRANT RD RR 3</p> <p data-bbox="805 789 1425 848">AddressLine2 Place rural route delivery information on a separate address line. For example,</p> <p data-bbox="971 873 1130 932">36 GRANT RD RR 3</p>
Place delivery office info in	<p data-bbox="805 1031 1425 1089">Specifies where to place station information. An example of an address with station information is:</p> <p data-bbox="805 1108 1032 1167">PO BOX 8625 STN A ST. JOHN'S NL</p> <p data-bbox="805 1188 1425 1278">Same as input Place station information in the same location as it is in the input address. Default.</p> <p data-bbox="805 1304 1425 1394">AddressLine1 Place station information on the same line as the address, after the address information. For example,</p> <p data-bbox="984 1419 1214 1446">PO BOX 8625 STN A</p> <p data-bbox="805 1478 1425 1537">AddressLine2 Place station information on a separate address line. For example,</p> <p data-bbox="984 1562 1143 1621">PO BOX 8625 STN A</p>

Option Name	Description						
Dual address logic	<p data-bbox="805 380 1414 495">Specifies whether Validate Address should return a street match or a PO Box/non-civic match when the address contains both civic and non-civic information. One of the following:</p> <table data-bbox="805 512 1386 699"> <tr> <td data-bbox="805 512 1024 539">Use default option</td> <td data-bbox="1062 512 1386 573">Use DualAddressLogic Global Option. Default.</td> </tr> <tr> <td data-bbox="805 590 976 617">PO Box Match</td> <td data-bbox="1062 590 1333 651">Match to PO Box or other non-street data.</td> </tr> <tr> <td data-bbox="805 667 959 695">Street Match</td> <td data-bbox="1062 667 1227 695">Match to street.</td> </tr> </table> <p data-bbox="805 724 1304 751">For example, given the following input address:</p> <p data-bbox="805 772 1122 896">AddressLine1: 36 GRANT RD AddressLine2: RR 4 City: ANTIGONISH StateProvince: NS</p> <p data-bbox="805 919 1341 947">Validate Address would return one of the following:</p> <ul data-bbox="805 968 1425 1415" style="list-style-type: none"> <li data-bbox="805 968 1425 1192">• If Dual address logic is set to <code>Street Match</code>, Validate Address returns the following: AddressLine1: 36 GRANT RD AddressLine2: RR 3 City: ANTIGONISH StateProvince: NS PostalCode: B2G 2L1 <li data-bbox="805 1220 1425 1415">• If Dual address logic is set to <code>PO Box Match</code>, Validate Address returns the following: AddressLine1: RR 4 City: ANTIGONISH StateProvince: NS PostalCode: B2G 2L2 <p data-bbox="805 1457 1425 1543">The address data that is not used to standardize the address is returned in the AdditionalInputData field. For more information, see Output Data Options on page 627.</p>	Use default option	Use DualAddressLogic Global Option. Default.	PO Box Match	Match to PO Box or other non-street data.	Street Match	Match to street.
Use default option	Use DualAddressLogic Global Option. Default.						
PO Box Match	Match to PO Box or other non-street data.						
Street Match	Match to street.						

SERP Processing

Validate Address allows for Software and Evaluation Recognition Processing (SERP). SERP processing enables you to qualify for Canada Post® postal discounts. Validate Address returns PoCAD data, which improves accuracy for house number and apartment data.

Note: You can return PoCAD data in batch mode only. If you try to return PoCAD data in real time, Validate Address will return with an error.

When you use SERP Certified™ processing, Validate Address generates a Canada Post SERP Statement of Accuracy. This form must be given to Canada Post along with the mailing to qualify for certain discounts. The form contains information about the software you used for SERP processing, information about your name-and-address list, information about your output file, information about the mailer, and other statistics about your mailing. For detailed information about Canada Post Address Accuracy Statement, see

<http://www.canadapost.ca/cpo/mc/business/productservices/atoz/addressaccuracy.jsf>.

To run Validate Address in SERP Certified™ mode, follow these steps:

1. Validate Address must be in SERP Certified™ mode. If **(Not SERP Certified)** appears at the top of the window, click the **Enable SERP settings** button. The **Configure SERP** box will appear.
2. Click **Configure SERP**. The **SERP Report Fields** dialog box appears.
3. Type your merchant **CPC number**.
4. Type the mailer **Name, Address, and City, State, ZIP**.
5. Click **OK**.
6. In Enterprise Designer, drag **SERPReport** from the Reports pallet to the canvas.

Note: You do not need to draw a connector between the Validate Address stage and the CASS3553 report.

7. Double-click the **SERPReport** icon on the canvas.
8. On the **Stages** tab, ensure that the **Validate Address** check box is checked. Note that if you have renamed the Validate Address stage to something else, you should check the box with the name you have given the address validation stage.
9. On the **Parameters** tab, select the format for the report. You can create the report in PDF, HTML, or plain text format. PDF format is the default.
10. Click **OK**.

Obtaining SERP Return Codes

SERP return codes indicate the quality of the input address as determined by the Canada Post's Software Evaluation and Recognition Program regulations.

To obtain SERP return codes, on the **Output Data** tab, select the **Include postal information** check box. This will return a variety of data about the address, including the SERP return codes. For information on the specific data that this option will return, see [Postal Data Output](#) on page 663.

SERP return codes are provided in the following output field.

Table 103: SERP Return Code Output

Field Name	Description
CanadianSERPCode	<p>Validation/correction return code (Canadian addresses only):</p> <p>V The input was valid. Canada Post defines a "valid" address as an address that meets all the following requirements:</p> <p style="padding-left: 40px;">Note: There are exceptions. For further information, contact the CPC.</p> <ul style="list-style-type: none"> • The address must contain all required components as found in CPC's Postal Code Data Files. • The address must provide an exact match on all components for only one address in CPC's Postal Code Data Files, allowing for acceptable alternate words and names listed in the CPC Postal Code Data Files. • Address components must be in a form that allows recognition without ambiguity. Certain components may require "qualifiers" to identify them. For instance, a Route Service address requires the key words "Rural Route" or "RR" for differentiation from a "Suburban Service" or "SS" address with the same number. <p>I The input was invalid. An "invalid" address is one that does not meet CPC requirements for a valid address (see above). Examples of this include address components that are missing, invalid, or inconsistent.</p> <p>C The input was correctable. A "correctable" address is one that can be corrected to match one, and only one, address.</p> <p>N The input was non-correctable. A "non-correctable" address is one that could be corrected a number of different ways such that Validate Address cannot identify a single correct version.</p> <p>F The input address was foreign (outside of Canada).</p>

International Address Options

Addresses outside of the U.S. and Canada are referred to as "international" addresses. The following options control international address processing:

Option Name	Description
Enable international address processing	<p>Specifies whether to process international addresses (addresses outside the U.S. and Canada). If you enable international address processing Validate Address will attempt to validate international addresses. If you disable international address processing, international addresses will fail, meaning they is returned with an "F" in the Status output field. The output field Status.Code will say "DisabledCoder." If you are not licensed for international address processing you must disable international address processing in order for your jobs to complete successfully, regardless of whether or not they contain international addresses.</p> <p>Note: You must have a valid license for international address processing to successfully process international addresses. If you enable international address processing but are not licensed for this feature, or your license has expired, your entire job will fail.</p>
Database	<p>Specifies which database you want to use for validating international addresses. To specify a database for international address validation, select a database in the Database drop-down list. Only databases that have been defined in the INTL Database Resources panel in the Management Console are available.</p>

Option Name	Description
International city and street searching	<p>By default, Validate Address provides a balance of good address matching accuracy with good performance. If you are willing to trade matching accuracy for faster performance, use the International city and street searching field to increase processing speed. When you do this, some accuracy is lost. This option only controls performance for addresses outside the U.S. and Canada. This setting affects a small percentage of records, mostly addresses in the U.K. There is no performance control for U.S. and Canadian address processing.</p> <p>If you use Get Candidate Addresses, the candidate addresses returned by Get Candidate Addresses may differ from the multiple matches returned by Validate Address if you set the performance tuning option for international addresses to any value other than 100.</p> <p>To control performance for addresses outside the U.S. and Canada, use the International city and street searching slider. To increase matching accuracy, move the slider to the right. A value of 100 results in the greatest accuracy. To increase processing speed, move the slider to the left. A value of 0 results in the greatest processing speed.</p>
Search address lines on fail	<p>This option enables Validate Address to search the AddressLine input fields for the city, state/province, postal code, and country when the address cannot be matched using the values in the City, StateProvince, and PostalCode input fields.</p> <p>Consider enabling this option if your input addresses have the city, state/province, and postal code information in the AddressLine fields.</p> <p>Consider disabling this option if your input addresses use the City, State/Province and PostalCode fields. If you enable this option and these fields are used, there is an increased possibility that Validate Address will fail to correct values in these fields (for example a misspelled city name).</p>

Output

The output from Validate Address contains different information depending on the output categories you select.

Standard Address Output

Standard address output consists of four lines of the address which correspond to how the address would appear on an address label. City, state/province, postal code, and other data is also included in standard address output. Standard address output is returned for validated addresses if you select the **Include a standard address** check box. Standard address fields are always returned for addresses that could not be validated. For non-validated addresses, the standard address output fields contain the address as it appeared in the input ("pass through" data). If you want addresses to be standardized according to postal authority standards when validation fails, select the **Include normalized data when no match is found** check box.

Table 104: Standard Address Output

Field Name	Description
AdditionalInputData	Input data not used by the address validation process. For more information, see About Additional Input Data .
AddressLine1	If the address was validated, the first line of the validated and standardized address. If the address could not be validated, the first line of the input address without any changes.
AddressLine2	If the address was validated, the second line of the validated and standardized address. If the address could not be validated, the second line of the input address without any changes.
City	The validated city name.
Country	The country in the format determined by what you selected in Country format: <ul style="list-style-type: none"> • ISO Code • UPU Code • English • French • German • Spanish
FirmName	The validated firm or company name.

Field Name	Description
PostalCode	The validated ZIP Code™ or postal code.
PostalCode.AddOn	The 4-digit add-on part of the ZIP Code™. For example, in the ZIP Code™ 60655-1844, 1844 is the 4-digit add-on. (U.S. addresses only.)
PostalCode.Base	The 5-digit ZIP Code™; for example 20706 (U.S. addresses only).
StateProvince	The validated state/province or its abbreviated value.
USUrbanName	The validated urbanization name. (U.S. addresses only.) This is used primarily for Puerto Rico addresses.
DefaultValidPostalCode	<p>This field is generated only for Canadian addresses.</p> <p>A value of Y indicates that the record is non-correctable (VN) type. In such cases, you have the option of lowering the output Confidence score for the record. To do this, select the Switch default valid postal code confidence check-box in the Input Options. For more information, see section Canadian Address Options on page 648.</p> <p>Note: For all other records, the field value is blank.</p>

Parsed Address Elements Output

Output addresses are formatted in the parsed address format if you select the **Include matched address elements** check box. If you want Validate Address to return formatted data in the Parsed Address format when validation fails (that is, a normalized address), select the **Return normalized data when no match is found** check box.

Note: If you always want return parsed input data returned regardless of whether or not validation is successful, select **Include standardized input address elements**. For more information, see [Parsed Input](#) on page 732.

Table 105: Parsed Address Output

Field Name	Description
AdditionalInputData	Input data not used by Validate Address. For more information, see Additional Input Data on page 690.
AdditionalInputData.Base	Input data that was not output to the standardized address by Validate Address. For more information, see Additional Input Data on page 690.
AdditionalInputData.Unmatched	Input data passed to the matcher but not used by Validate Address for validation. For more information, see Additional Input Data on page 690.
ApartmentLabel	Apartment designator (such as STE or APT), for example: 123 E Main St APT 3
ApartmentLabel2	Secondary apartment designator, for example: 123 E Main St APT 3, 4th Floor Note: In this release, this field will always be blank.
ApartmentNumber	Apartment number. For example: 123 E Main St APT 3
ApartmentNumber2	Secondary apartment number. For example: 123 E Main St APT 3, 4th Floor Note: In this release, this field will always be blank.
CanadianDeliveryInstallationAreaName	Delivery installation name (Canadian addresses only)
CanadianDeliveryInstallationQualifierName	Delivery installation qualifier (Canadian addresses only)
CanadianDeliveryInstallationType	Delivery installation type (Canadian addresses only)

Field Name	Description
HouseNumber	House number, for example: 123 E Main St Apt 3
LeadingDirectional	Leading directional, for example: 123 E Main St Apt 3
POBox	Post office box number. If the address is a rural route address, the rural route box number will appear here.
PrivateMailbox	Private mailbox indicator.
PrivateMailbox.Type	The type of private mailbox. Possible values include: <ul style="list-style-type: none"> • Standard • Non-Standard <p>Note: This replaces PrivateMailboxType (no period in field name). Please modify your API calls accordingly.</p>
RRHC	Rural Route/Highway Contract indicator
StreetName	Street name, for example: 123 E Main St Apt 3
StreetSuffix	Street suffix, for example: 123 E Main St Apt 3
TrailingDirectional	Trailing directional, for example: 123 Pennsylvania Ave NW

Parsed Input

The output can include the input address in parsed form. This type of output is referred to as "parsed input." Parsed input fields contain the address data that was used as input regardless of whether or not Validate Address validated the address. Parsed input is different from the "parsed address elements" output in that parsed address elements contain the validated address if the address could be validated, and, optionally, the input address if the address could not be validated. Parsed input always contains the input address regardless of whether or not Validate Address validated the address.

To include parsed input fields in the output, select the **Return parsed input data** check box.

Table 106: Parsed Input

Field Name	Description
ApartmentLabel.Input	Apartment designator (such as STE or APT), for example: 123 E Main St APT 3
ApartmentNumber.Input	Apartment number, for example: 123 E Main St APT 3
CanadianDeliveryInstallationAreaName.Input	Delivery installation name (Canadian addresses only)
CanadianDeliveryInstallationQualifierName.Input	Delivery installation qualifier (Canadian addresses only)
CanadianDeliveryInstallationType.Input	Delivery installation type (Canadian addresses only)
City.Input	Validated city name
Country.Input	Country. Format is determined by what you selected in Country format: <ul style="list-style-type: none"> • ISO Code • UPU Code • English • French • German • Spanish
FirmName.Input	The validated firm or company name
HouseNumber.Input	House number, for example: 123 E Main St Apt 3
LeadingDirectional.Input	Leading directional, for example: 123 E Main St Apt 3

Field Name	Description
POBox.Input	Post office box number. If the address is a rural route address, the rural route box number will appear here.
PostalCode.Input	Validated postal code. For U.S. addresses, this is the ZIP Code.
PrivateMailbox.Input	Private mailbox indicator
PrivateMailbox.Type.Input	The type of private mailbox. Possible values include: <ul style="list-style-type: none"> • Standard • Non-Standard
RRHC.Input	Rural Route/Highway Contract indicator
StateProvince.Input	Validated state or province name
StreetName.Input	Street name, for example: 123 E Main St Apt 3
StreetSuffix.Input	Street suffix, for example: 123 E Main St Apt 3
TrailingDirectional.Input	Trailing directional, for example: 123 Pennsylvania Ave NW
USUrbanName.Input	USPS® urbanization name

Postal Data Output

If you select **Include postal information** then the following fields are returned in the output.

Table 107: Postal Data Output

Field Name	Description
IntHexaviaCode	For addresses in France only, a numeric code that represents the street. For information about Hexavia codes, see www.laposte.fr .
IntINSEECODE	For addresses in France only, a numeric code that represents the city. For a listing of INSEE codes, see www.insee.fr .
PostalBarCode	The two-digit delivery point portion of the delivery point barcode (U.S. addresses only). For more information, see Creating Delivery Point Barcodes on page 631.
USAltAddr	Indicates whether or not alternate address matching logic was used, and if so which logic was used (U.S. addresses only). One of the following: <ul style="list-style-type: none"> null No alternate address scheme used. D Delivery point alternate logic was used. E Enhanced highrise alternate match logic was used. S Small town default logic was used. U Unique ZIP Code logic was used.
USBCCheckDigit	Check-digit portion of the 11-digit delivery point barcode (U.S. addresses only). For more information, see Creating Delivery Point Barcodes on page 631.
USCarrierRouteCode	Carrier route code (U.S. addresses only). For more information, see Obtaining Carrier Route Codes on page 631.
USCongressionalDistrict	Congressional district (U.S. addresses only). For more information, see Obtaining Congressional Districts on page 630.
USCountyName	County name (U.S. addresses only). For more information, see Obtaining County Names on page 630.

Field Name	Description
USFinanceNumber	The finance number in which the address resides (U.S. addresses only). The finance number is a number assigned by the USPS to an area that covers multiple ZIP Codes. An address is validated only if its finance number matches the finance number of the candidate address in the U.S. Database.
USFIPSCountyNumber	FIPS (Federal Information Processing Standards) county number (U.S. addresses only). For more information, see Obtaining FIPS County Numbers on page 630.
USLACS	<p>Indicates whether or not the address is a candidate for LACS^{Link} conversion (U.S. addresses only). One of the following:</p> <p>Y Yes, the address is a candidate for LACS^{Link} processing. If LACS^{Link} is enabled, an attempt is made to convert the address using the LACS^{Link} database. If the conversion attempt is successful, the output address is the new address obtained from the LACS^{Link} database. If the attempt is not successful, the address will not be converted.</p> <p>N No, the address is not a candidate for LACS^{Link} processing. LACS^{Link} processing may still be attempted if LACS^{Link} processing is requested, the LACS^{Link} database is installed, and one of the following is true:</p> <ul style="list-style-type: none"> • The address matches to a Rural Route address and the RecordType.Default field returns a Y. • The input address could not be matched to any address in the U.S. Postal Database (Failures due to multiple matches are not LACS^{Link} candidates.)
USLastLineNumber	<p>A six-character alphanumeric value that groups together ZIP Codes that share the same primary city. For example, addresses with the following two last lines would have the same last line number:</p> <p>Chantilly VA 20151 Chantilly VA 20152</p>

Result Indicators

Result indicators provide information about the kinds of processing performed on an address. There are two types of result indicators:

Record-Level Result Indicators

Record-level result indicators provide data about the results of Validate Address processing for each record, such as the success or failure of the match attempt, which coder processed the address,

and other details. The following table lists the record-level result indicators returned by Validate Address.

Table 108: Record Level Indicators

Field Name	Description
AddressFormat	<p>The type of address data being returned:</p> <p>F French format (for example: 123 Rue Main)</p> <p>E English format (for example: 123 Main St)</p>
Confidence	<p>The level of confidence assigned to the address being returned. Range is from zero (0) to 100; zero indicates failure, 100 indicates a very high level of confidence that the match results are correct. For multiple matches, the confidence level is 0. For details about how this number is calculated, see Introduction to the Validate Address Confidence Algorithm.</p>
CouldNotValidate	<p>If no match was found, which address component could not be validated:</p> <ul style="list-style-type: none"> • ApartmentNumber • HouseNumber • StreetName • PostalCode • City • Directional • StreetSuffix • Firm • POBoxNumber • RuralRoute <p>Note: More than one component may be returned, in a comma-separated list.</p>

Field Name	Description
CountryLevel	<p>The category of address matching available. This is always "A" for U.S. and Canadian addresses. One of the following:</p> <p>A The address is in a country for which there is highly detailed postal data available. Addresses in this match level can have the following address elements validated and corrected, and added if missing from the input:</p> <ul style="list-style-type: none">• Postal code• City name• State/county name• Street address elements• Country name <p>B The address is in a country for which there is a medium level of postal data available. Addresses in this match level can have the following address elements validated and corrected, and added if missing from the input:</p> <ul style="list-style-type: none">• Postal code• City name• State/county name• Country name <p>C The address is in a country for which the postal data is least detailed. Addresses in this match level can have the following actions performed on them:</p> <ul style="list-style-type: none">• Validate and correct country name (cannot supply missing country name)• Validate the format of the postal code (cannot supply missing postal code or validate the code)

Field Name	Description
MatchScore	<p>MatchScore provides an indication of the degree to which the output address is correct. It is significantly different from Confidence in that Confidence indicates how much the input address changed to obtain a match, whereas the meaning of Match Score varies between U.S. and non-U.S. addresses.</p> <p>For U.S. addresses, MatchScore is a one-digit score on a scale of 0 to 9 that reflects the closeness of the street-name match (after transformations by Validate Address, if any). Zero indicates an exact match and 9 indicates the least likely match. If no match was found, this field is blank.</p> <p>For non-U.S. and non-Canadian addresses, MatchScore is a five-digit score, with a maximum value of 00999. Higher numbers indicates a closer match.</p> <p>This field does not apply to Canadian addresses.</p> <p>Note that you cannot equate match scores from U.S. addresses with those of non-U.S. addresses. For example, a match score of 4 for a U.S address does not indicate the same level of match as a 00004 for a non-U.S. address.</p> <p>Note: The Validate Address and Advanced Matching Module components both use the MatchScore field. The MatchScore field value in the output of a dataflow is determined by the last stage to modify the value before it is sent to an output stage. If you have a dataflow that contains Validate Address and Advanced Matching Module components and you want to see the MatchScore field output for each stage, use a Transformer stage to copy the MatchScore value to another field. For example, Validate Address produces an output field called MatchScore and then a Transformer stage copies the MatchScore field from Validate Address to a field called AddressMatchScore. When the matcher stage runs it populates the MatchScore field with the value from the matcher and passes through the AddressMatchScore value from Validate Address.</p>
MultimatchCount	If multiple matches were found, indicates the number of records that are possible matches.
MultipleMatches	<p>Indicates which address component had multiple matches, if multiple matches were found:</p> <ul style="list-style-type: none"> • Firm • LeadingDirectional • PostalCode • StreetName • StreetSuffix • TrailingDirectional • Urbanization <p>Note: More than one component may be returned, in a comma-separated list.</p>

Field Name	Description						
ProbableCorrectness	<p>Indicates the accuracy of a match on the scale of 0 to 9. The result can be:</p> <ul style="list-style-type: none"> • <i>Blank</i> - No Match Found • 0 - Most likely to be correct (Exact Match) • 1-8 - Intermediate probability of being correct • 9 - Match least likely to be correct 						
ProcessedBy	<p>Which address coder processed the address:</p> <table> <tr> <td>USA</td> <td>U.S. address coder</td> </tr> <tr> <td>CAN</td> <td>Canadian address coder</td> </tr> <tr> <td>INT</td> <td>International address coder</td> </tr> </table>	USA	U.S. address coder	CAN	Canadian address coder	INT	International address coder
USA	U.S. address coder						
CAN	Canadian address coder						
INT	International address coder						
RecordType	<p>Type of address record, as defined by U.S. and Canadian postal authorities (supported for U.S. and Canadian addresses only):</p> <ul style="list-style-type: none"> • FirmRecord • GeneralDelivery • HighRise • PostOfficeBox • RRHighwayContract • Normal 						
RecordType.Default	<p>Code indicating the "default" match:</p> <table> <tr> <td>Y</td> <td>The address matches a default record.</td> </tr> <tr> <td>null</td> <td>The address does not match a default record.</td> </tr> </table>	Y	The address matches a default record.	null	The address does not match a default record.		
Y	The address matches a default record.						
null	The address does not match a default record.						
Status	<p>Reports the success or failure of the match attempt. For multiple matches, this field is "F" for all the possible matches.</p> <table> <tr> <td>null</td> <td>Success</td> </tr> <tr> <td>F</td> <td>Failure</td> </tr> </table>	null	Success	F	Failure		
null	Success						
F	Failure						

Field Name	Description
Status.Code	Reason for failure, if there is one. For multiple matches, all possible matches is "MultipleMatchesFound." <ul style="list-style-type: none"> • DisabledCoder • InsufficientInputData • MultipleMatchesFound • UnableToValidate
Status.Description	Description of the problem, if there is one. Possible Multiple Addresses Found This value will appear if Status.Code=MultipleMatchesFound. Address Not Found This value will appear if Status.Code=UnableToValidate. PerformUSProcessing disabled This value will appear if Status.Code=DisabledCoder. PerformCanadianProcessing disabled This value will appear if Status.Code=DisabledCoder. PerformInternationalProcessing disabled This value will appear if Status.Code=DisabledCoder.

Field-Level Result Indicators

Field-level result indicators describe how Validate Address handled each address element. Field-level result indicators are returned in the qualifier "Result". For example, the field-level result indicator for HouseNumber is contained in **HouseNumber.Result**.

To enable field-level result indicators, check the **Include result codes for individual fields** check box. For more information, see [Output Data Options](#) on page 627.

The following table lists the field-level result indicators. If a particular field does not apply to an address, the result indicator may be blank.

Table 109: Field-Level Result Indicators

Field Name	Description
AddressRecord.Result	<p>These result codes apply to international addresses only.</p> <ul style="list-style-type: none"> M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. S Standardized. This option includes any standard abbreviations. U Unmatched. V Validated. The data was confirmed correct and remained unchanged from input.
ApartmentLabel.Result	<ul style="list-style-type: none"> A Appended. The field was added to a blank input field. U.S. and Canadian addresses only. C Corrected. U.S. and Canadian addresses only. D Dropped. The field provided on input was removed. U.S. and Canadian addresses only. For more information, see About Additional Input Data. F Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses. P Pass-through. The data was not used in the validation process, but it was preserved in the output. U.S. and Canadian addresses only. R The apartment label is required but is missing from the input address. U.S. addresses only. S Standardized. This option includes any standard abbreviations. U Unmatched. Does not apply to Canadian addresses. V Validated. The data was confirmed correct and remained unchanged from input.

Field Name	Description
ApartmentNumber.Result	<p>A Appended. The field was added to a blank input field. U.S. and Canadian addresses only.</p> <p>C Corrected. Canadian addresses only.</p> <p>D Dropped. The field provided on input was removed. U.S. addresses only. For more information, see About Additional Input Data.</p> <p>F Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.</p> <p>P Pass-through. The data was not used in the validation process, but it was preserved in the output. U.S. addresses that are an EWS match will have a value of P. U.S. and Canadian addresses only.</p> <p>R The apartment number is required but is missing from the input address. U.S. addresses only.</p> <p>S Standardized. This option includes any standard abbreviations. Does not apply to U.S. addresses.</p> <p>U Unmatched.</p> <p>V Validated. The data was confirmed correct and remained unchanged from input.</p>

Field Name	Description
City.Result	<p>A Appended. The field was added to a blank input field. U.S. and Canadian addresses only.</p> <p>C Corrected. U.S. and Canadian addresses only.</p> <p>F Hyphens missing or punctuation errors. Canadian addresses only.</p> <p>M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. Does not apply to U.S. or Canadian addresses.</p> <p>P Pass-through. The data was not used in the validation process, but it was preserved in the output.</p> <p>R The city is required but is missing from the input address. U.S. addresses only.</p> <p>S Standardized. This option includes any standard abbreviations. Does not apply to U.S. addresses.</p> <p>U Unmatched. Does not apply to Canadian addresses.</p> <p>V Validated. The data was confirmed correct and remained unchanged from input.</p>
Country.Result	<p>These result codes do not apply to U.S. or Canadian addresses.</p> <p>M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field.</p> <p>S Standardized. This option includes any standard abbreviations.</p> <p>U Unmatched.</p> <p>V Validated. The data was confirmed correct and remained unchanged from input.</p>

Field Name	Description
FirmName.Result	C Corrected. U.S. addresses only.
	P Pass-through. The data was not used in the validation process, but it was preserved in the output. U.S. and Canadian addresses only.
	U Unmatched. U.S. and Canadian addresses only.
	V Validated. The data was confirmed correct and remained unchanged from input. U.S. addresses only.
HouseNumber.Result	A Appended. The field was added to a blank input field. Canadian addresses only.
	C Corrected. Canadian addresses only.
	D Dropped. The field provided on input was removed. U.S. addresses only. For more information, see About Additional Input Data .
	F Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	O Out of range. Does not apply to U.S. or Canadian addresses.
	P Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only.
	R The house number is required but is missing from the input address. Canadian addresses only.
	S Standardized. This option includes any standard abbreviations. Does not apply to U.S. or Canadian addresses.
	U Unmatched.
	V Validated. The data was confirmed correct and remained unchanged from input.

Field Name	Description
LeadingDirectional.Result	<p>A Appended. The field was added to a blank input field. U.S. and Canadian addresses only.</p> <p>C Corrected. Non-blank input was corrected to a non-blank value. U.S. addresses only.</p> <p>D Dropped. The field provided on input was removed. U.S. addresses only. For more information, see About Additional Input Data.</p> <p>F Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.</p> <p>M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only.</p> <p>P Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only.</p> <p>S Standardized. This option includes any standard abbreviations.</p> <p>U Unmatched.</p> <p>V Validated. The data was confirmed correct and remained unchanged from input. Does not apply to Canadian addresses.</p>

Field Name	Description
POBox.Result	<p>A Appended. The field was added to a blank input field. Canadian addresses only.</p> <p>C Corrected. Canadian addresses only.</p> <p>D Dropped. The field provided on input was removed. U.S. addresses only. For more information, see About Additional Input Data.</p> <p>F Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.</p> <p>M Multiple matches. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only.</p> <p>P Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only.</p> <p>R The P.O. Box number is required but is missing from the input address. U.S. addresses only.</p> <p>S Standardized. This option includes any standard abbreviations.</p> <p>U Unmatched.</p> <p>V Validated. The data was confirmed correct and remained unchanged from input.</p> <p>Blank Not applicable.</p>

Field Name	Description
PostalCode.Result	<p>A Appended. The field was added to a blank input field. U.S. and Canadian addresses only.</p> <p>C Corrected. U.S. and Canadian addresses only.</p> <p>F Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.</p> <p>M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. Does not apply to Canadian addresses.</p> <p>P Pass-through. The data was not used in the validation process, but it was preserved in the output. Does not apply to U.S. addresses.</p> <p>R The postal code is required but is missing from the input address. U.S. addresses only.</p> <p>S Standardized. This option includes any standard abbreviations. Does not apply to U.S. or Canadian addresses.</p> <p>U Unmatched. For example, if the street name does not match the postal code, both StreetName.Result and PostalCode.Result will contain U.</p> <p>V Validated. The data was confirmed correct and remained unchanged from input.</p>
PostalCodeCity.Result	<p>These result codes apply to international addresses only.</p> <p>M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field.</p> <p>P Pass-through. The data was not used in the validation process, but it was preserved in the output.</p> <p>S Standardized. This option includes any standard abbreviations.</p> <p>U Unmatched.</p> <p>V Validated. The data was confirmed correct and remained unchanged from input.</p>

Field Name	Description
PostalCode.Source	<p>These result codes apply to U.S. addresses only.</p> <p>FinanceNumber The ZIP Code™ in the input was verified by using USPS® Finance Number groupings.</p> <p>ZIPMOVE The ZIP Code™ in the input address was corrected because the USPS® redrew ZIP Code™ boundaries and the address is now in a different ZIP Code™.</p>
PostalCode.Type	<p>P The ZIP Code™ contains only PO Box addresses. U.S. addresses only.</p> <p>U The ZIP Code™ is a unique ZIP Code™ assigned to a specific company or location. U.S. addresses only.</p> <p>M The ZIP Code™ is for military addresses. U.S. addresses only.</p> <p>null The ZIP Code™ is a standard ZIP Code™.</p>
RRHC.Result	<p>C Corrected. Canadian addresses only.</p> <p>D Dropped. The field provided on input was removed. U.S. addresses only. For more information, see About Additional Input Data.</p> <p>M Multiple matches. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only.</p> <p>P Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only.</p> <p>R The rural route/highway contract is required but is missing from the input address. U.S. addresses only.</p> <p>S Standardized. This option includes any standard abbreviations. U.S. and Canadian addresses only.</p> <p>U Unmatched. U.S. and Canadian addresses only.</p> <p>V Validated. The data was confirmed correct and remained unchanged from input. U.S. and Canadian addresses only.</p>

Field Name	Description
RRHC.Type	<p>These result codes apply to U.S. addresses only.</p> <p>HC The address is a Highway Contract address.</p> <p>RR The address is a Rural Route address.</p>
StateProvince.Result	<p>A Appended. The field was added to a blank input field. U.S. and Canadian addresses only.</p> <p>C Corrected. U.S. addresses only.</p> <p>M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. Does not apply to U.S. or Canadian addresses.</p> <p>P Pass-through. The data was not used in the validation process, but it was preserved in the output.</p> <p>R The state is required but is missing from the input address. U.S. addresses only.</p> <p>S Standardized. This option includes any standard abbreviations. Does not apply to U.S. addresses.</p> <p>U Unmatched. Does not apply to Canadian addresses.</p> <p>V Validated. The data was confirmed correct and remained unchanged from input.</p>
Street.Result	<p>These result codes apply to international addresses only.</p> <p>M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field.</p> <p>P Pass-through. The data was not used in the validation process, but it was preserved in the output.</p> <p>R Street corrected. House number is out of range. Applies to French, UK, and Japanese records only.</p> <p>S Standardized. This option includes any standard abbreviations.</p> <p>U Unmatched.</p> <p>V Validated. The data was confirmed correct and remained unchanged from input.</p>

Field Name	Description
StreetName.AbbreviatedAlias.Result	<p>Indicates the result of abbreviated alias processing. One of the following:</p> <ul style="list-style-type: none"> null No abbreviated alias processing attempted. B The StreetName field contains the base street name. L The standardized address length is less than 31 characters so the StreetName field contains the base name. N No abbreviated alias found. Y An abbreviated alias was found for input address. The StreetName field contains the abbreviated alias.
StreetName.Alias.Type	<p>This result code applies to U.S. addresses only.</p> <p>Note: In previous releases this field was named StreetName.AliasType with no "." between "Alias" and "Type." This old name is obsolete. Please update your processes to use the new name StreetName.Alias.Type.</p> <ul style="list-style-type: none"> Abbreviated The alias is an abbreviation of the street name. For example, HARTS-NM RD is an abbreviated alias for HARTSVILLE NEW MARLBORO RD. Changed There has been an official street name change and the alias reflects the new name. For example if SHINGLE BROOK RD is changed to CANNING DR, then CANNING DR would be a changed alias type. Other The street alias is made up of other names for the street or common abbreviations of the street. Preferred The street alias is the locally preferred alias. For example, a street is named "South Shore Dr." because it runs along the southern shore of a lake, not because it is south of a municipal demarcation line. So, "South" is not a predirectional in this case and should not be shorted to "S". So, "South Shore Dr." would be the preferred alias.

Field Name	Description
StreetName.PreferredAlias.Result	<p>Indicates the result of preferred alias processing. One of the following:</p> <ul style="list-style-type: none"> null No preferred alias processing attempted. A Preferred alias processing was not attempted because the input address matched to an alias. Preferred alias processing is only attempted for base addresses. N No preferred alias found. Y A preferred alias was found for the input address. The StreetName field contains the preferred alias.
StreetName.Result	<ul style="list-style-type: none"> A Appended. The field was added to a blank input field. Canadian addresses only. C Corrected. U.S. and Canadian addresses only. D Dropped. The field provided on input was removed. U.S. addresses only. For more information, see About Additional Input Data. F Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses. M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only. P Pass-through. The data was not used in the validation process, but it was preserved in the output. Does not apply to U.S. addresses. S Standardized. This option includes any standard abbreviations. U.S. and Canadian addresses only. U Unmatched. V Validated. The data was confirmed correct and remained unchanged from input.

Field Name	Description
StreetSuffix.Result	<p>A Appended. The field was added to a blank input field. U.S. and Canadian addresses only.</p> <p>C Corrected. U.S. and Canadian addresses only.</p> <p>D Dropped. The field provided on input was removed. U.S. and Canadian addresses only. For more information, see About Additional Input Data.</p> <p>F Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.</p> <p>M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only.</p> <p>P Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only.</p> <p>S Standardized. This option includes any standard abbreviations.</p> <p>U Unmatched. Does not apply to U.S. addresses.</p> <p>V Validated. The data was confirmed correct and remained unchanged from input.</p>

Field Name	Description
TrailingDirectional.Result	<p>A Appended. The field was added to a blank input field. U.S. and Canadian addresses only.</p> <p>C Corrected. U.S. and Canadian addresses only.</p> <p>D Dropped. The field provided on input was removed. U.S. and Canadian addresses only. For more information, see About Additional Input Data.</p> <p>F Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.</p> <p>M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only.</p> <p>P Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only.</p> <p>S Standardized. This option includes any standard abbreviations.</p> <p>U Unmatched. Does not apply to Canadian addresses.</p> <p>V Validated. The data was confirmed correct and remained unchanged from input.</p>
USUrbanName.Result	<p>These result codes apply to U.S. addresses only.</p> <p>A Appended. The field was added to a blank input field.</p> <p>C Corrected.</p> <p>M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field.</p> <p>U Unmatched.</p> <p>V Validated. The data was confirmed correct and remained unchanged from input.</p>

Output from Options

Validate Address returns additional data depending on the options you select. For information on the output generated by each option, see the options listed in the following sections:

Enhanced Line of Travel Output

Enhanced Line of Travel processing produces the following output.

Field Name	Description
USLOTCode	<p>Line of Travel sequence code and an indicator denoting USPS® LOT sequence. This field is in the format nnnnY where:</p> <p>nnnn The four-digit LOT code.</p> <p>Y One of the following:</p> <ul style="list-style-type: none"> • A—Ascending LOT sequence • D—Descending LOT sequence
USLOTHex	A hexadecimal value that allows you to sort your file in ascending order only. The hexadecimal values range from 0 to FF ascending, then FF through 0 descending.
USLOTSequene	A two-byte value used for final sortation in place of the DPC add-on. It consists of an uppercase letter followed by a digit 0 through 9. Values range from A0 (99 descending) through J9 (00 descending), and K0 (00 ascending) through T9 (99 ascending).

LACS^{Link} Output

Field Name	Description
USLACS	<p>Indicates whether or not the address is a candidate for LACS^{Link} conversion (U.S. addresses only). One of the following:</p> <p>Y Yes, the address is a candidate for LACS^{Link} processing. If LACS^{Link} is enabled, Validate Address will attempt to convert the address using the LACS^{Link} database. If the conversion attempt is successful, the output address is the new address obtained from the LACS^{Link} database. If the attempt is not successful, the address will not be converted.</p> <p>N No, the address is not a candidate for LACS^{Link} processing. LACS^{Link} processing may still be attempted if LACS^{Link} processing is requested, the LACS^{Link} database is installed, and one of the following is true:</p> <ul style="list-style-type: none"> • The address matches to a Rural Route address and the RecordType.Default field returns a Y. • The input address could not be matched to any address in the U.S. Postal Database (Failures due to multiple matches are not LACS^{Link} candidates.)
USLACS.ReturnCode	<p>Indicates the success or failure of LACS^{Link} processing. (U.S. addresses only.)</p> <p>A LACS^{Link} processing successful. Record matched through LACS^{Link} processing.</p> <p>00 LACS^{Link} processing failed. No matching record found during LACS^{Link} processing.</p> <p>09 LACS^{Link} processing matched the input address to an older highrise default address. The address has been converted. Rather than provide an imprecise address, LACS^{Link} processing does not provide a new address.</p> <p>14 LACS^{Link} processing failed. Match found during LACS^{Link} processing but conversion did not occur due to other USPS[®] regulations.</p> <p>92 LACS^{Link} processing successful. Record matched through LACS^{Link} processing. Unit number dropped on input.</p> <p>null LACS^{Link} did not process the record, or LACS^{Link} processing was not attempted.</p>

RDI Output

Field Name	Description
RDI	Return values indicating address type. B The address is a business address. R The address is a residential address. M The address is both a residential and a business address. null Not checked because the address did not code at a ZIP + 4 [®] level, or RDI™ was not performed.

DPV and CMRA Output

Field Name	Description
DPV	<p>Indicates the results of Delivery Point Validation (DPV) processing.</p> <p>Y DPV confirmed. Mail can be delivered to the address.</p> <p>N Mail cannot be delivered to the address.</p> <p>S The building number was validated but the unit number could not be confirmed. A building number is the primary address number for a building. A unit number is a number of a distinct mailing address within a building such as an apartment, suite, floor, and so on. For example, in this address 424 is the building number and 12 is the unit number:</p> <p>424 Washington Blvd. Apt. 12 Oak Park IL 60302 USA</p> <p>D The building number was validated but the unit number was missing from input. A building number is the primary address number for a building. A unit number is a number of a distinct mailing address within a building such as an apartment, suite, floor, and so on. For example, in this address 424 is the building number and 12 is the unit number:</p> <p>424 Washington Blvd. Apt. 12 Oak Park IL 60302 USA</p> <p>M The address matches multiple valid delivery points.</p> <p>U The address could not be confirmed because the address did not code at the ZIP + 4[®] level.</p> <p>V The address caused a false-positive violation.</p>
CMRA	<p>Indicates if the address is a Commercial Mail Receiving Agency (CMRA)</p> <p>Y Yes, the address is a CMRA.</p> <p>N No, the address is not a CMRA.</p> <p>U Unconfirmed.</p>

Field Name	Description
DPVFootnote	<p>DPV footnote codes.</p> <p>AA Input address matched to the ZIP + 4[®] file.</p> <p>A1 Input address not matched to the ZIP + 4[®] file.</p> <p>BB Input address matched to DPV (all components).</p> <p>CC Input address primary number matched to DPV but secondary number not match (present but not valid).</p> <p>F1 Input address is military; DPV bypassed.</p> <p>G1 Input address is general delivery; DPV bypassed.</p> <p>M1 Input address primary number missing.</p> <p>M3 Input address primary number invalid.</p> <p>N1 Input address primary number matched to DPV but high rise address missing secondary number.</p> <p>P1 Input address missing RR or HC Box number.</p> <p>P3 Input address missing PO, RR, or HC Box number</p> <p>RR Input address matched to CMRA.</p> <p>R1 Input address matched to CMRA but secondary number not present.</p> <p>R7 Input address matched to phantom carrier route R777 (not eligible for street delivery).</p> <p>U1 Input address is unique ZIP; DPV bypassed.</p>
DPVVacant	<p>Indicates whether the building is vacant (unoccupied for 90 days). One of the following:</p> <p>Y Yes, the building is vacant.</p> <p>N No, the building is not vacant.</p> <p>null The Determine if address is vacant option was not turned on.</p>
DPVNoStat	<p>Indicates whether the building is a "no stat" building and therefore unable to receive mail. One of the following:</p> <p>Y Yes, the building is a "no stat" building, which means the building is not receiving mail.</p> <p>N No, the building is not a "no stat" building, which means the building does receive mail.</p> <p>null The Determine if delivery point is active option was not turned on.</p>

Suite^{Link} Output

Field Name	Description
SuiteLinkReturnCode	<p>Indicates whether or not Validate Address corrected the secondary address information (U.S. addresses only). One of the following:</p> <p>A Validate Address corrected the secondary address information.</p> <p>00 Validate Address did not correct the secondary address information.</p> <p>null Suite^{Link} was not performed.</p> <p>XX Suite^{Link} processing encountered an error. For example, an error would occur if the Suite^{Link} database is expired.</p>
SuiteLinkMatchCode	<p>Provides additional information on the Suite^{Link} match attempt. (U.S. addresses only)</p> <p>A Validate Address corrected the secondary address information.</p> <p>B Validate Address did not correct the secondary address information. No additional detail about the match attempt is available.</p> <p>C The words in the FirmName field are all "noise" words. Noise words are defined by the USPS[®] and are ignored when attempting to match the firm name. Examples of noise words are "company" and "corporation". Validate Address is not able to correct secondary address information for firm names that consist entirely of noise words. For example "Company and Corporation" is all noise words.</p> <p>D The address is not a high-rise default address. Suite^{Link} matching is only done for high-rise default addresses. A high-rise default is a default to use when the address does not contain valid secondary information (the apartment number or apartment type is missing).</p> <p>E Suite^{Link} processing failed because the Suite^{Link} database is expired.</p> <p>null Suite^{Link} was not performed or there was an error.</p>

Field Name	Description
SuiteLinkFidelity	<p>Indicates how well Validate Address matched the firm name to the firm names in the Suite^{Link} database.</p> <p>1 The firm name matches the Suite^{Link} database exactly.</p> <p>2 Good match. All words in the firm name except one matched the firm name in the Suite^{Link} database.</p> <p>3 Poor match. More than one word in the firm name did not match the firm name in the Suite^{Link} database.</p> <p>null Suite^{Link} could not match the firm name, or was not performed, or there was an error.</p>

VeriMove Output

Field Name	Description
VeriMoveDataBlock	<p>Indicates whether or not Validate Address should return a 250-byte field containing input data to pass to VeriMove Express. This field contains the Detail Results Indicator data required by VeriMove. For more information about the contents of this field, see the VeriMove User's Guide. One of the following:</p> <p>Y Yes, return the field VeriMoveDataBlock.</p> <p>N No, do not return the field VeriMoveDataBlock.</p>

Additional Input Data

Some input data is ignored during the address standardization process. This extraneous data (sometimes referred to as "dropped data") is returned in the AdditionalInputData field. Some examples of dropped data include:

- Delivery instructions (for example, "Leave at back door")
- Phone numbers (for example, "555-135-8792")
- Attention lines (for example, "Attn: John Smith")

Data such as this is generally not embedded in an address. If it is embedded, the extraneous data can usually be identified and returned in the AdditionalInputData field.

Note: Dropped data from split indicia addresses is not returned. A split indicia address is one where a primary address is split between multiple address lines. For example, if the primary

address is "1 Green River Valley Rd" then the following would be a split indicia version of this address:

1 Green River
Valley Rd
01230

If there is more than one piece of dropped data in an address, each piece of data is separated by a semicolon and a space ("; ") for U.S. addresses and a space for addresses outside the U.S. The order of dropped data in `AdditionalInputData` is:

1. Care of, mail stop (U.S. addresses only)
2. Other extraneous data found on address lines
3. Entire unused data lines

For example, if this is the input address:

123 Main St C/O John Smith
Apt 5 Drop at back dock
jsmith@example.com
555-123-4567
05674

Then `AdditionalInputData` would contain:

C/O John Smith; Apt 5 Drop At Back Dock; 555-123-4567; Jsmith@example.com; 555-123-4567

Care of Data

For U.S. addresses only, "care of" data is returned in `AdditionalInputData`. The following addresses contain examples of "care of" data:

123 Main St C/O John Smith
Apt 5
05674

123 Main St
Apt 5 ATTN John Smith
05674

123 Main St Apt 5
MailStop 2
05674

Extraneous Data on Its Own Address Line

`Validate Address` returns extraneous data on its own address line for U.S. and Canadian addresses.

For U.S. addresses, `Validate Address` uses the first two non-blank address lines to perform address standardization, unless either the firm name extraction or urbanization code extraction options are enabled (see [Address Line Processing for U.S. Addresses](#) on page 626 for more information).

Data on other address lines is returned in `AdditionalInputData`. In the following address, "John Smith" would be returned in `AdditionalInputData` because it is in the third non-blank address line and `ValidateAddress` only uses the first two non-blank address lines for U.S. addresses.

```
123 Main St
Apt 5
John Smith
05674
```

If one of either of the first two non-blank address lines contains extraneous data, that data is returned in `AdditionalInputData`. For example, in the following addresses "John Smith" would be returned in `AdditionalAddressData`.

```
123 Main St
John Smith
05674
```

```
John Smith
123 Main St
05674
```

In the following address both "John Smith" and "Apt 5" would both be returned in `AdditionalInputData`. "John Smith" would be returned because it is extraneous data in one of the first two address lines and "Apt 5" would be returned because U.S. address data must be in the first two non-blank address lines.

```
John Smith
123 Main St
Apt 5
05674
```

Extraneous Data Within an Address Line

Extraneous data that is within an address line is returned in `AdditionalInputData`. For example, in the following addresses "John Smith" would be returned in `AdditionalInputData`.

```
123 Main St John Smith
05674
```

```
123 Main St Apt 5 John Smith
05674
```

```
123 Main St John Smith
Apt 5
05674
```

```
123 Main St
Apt 5 John Smith
05674
```

For U.S. addresses, only extraneous data at the end of the address line is returned in `AdditionalInputData`. Extraneous data that is not at the end of an address line is not returned for U.S. addresses. For example, in the following addresses "John Smith" is not returned.

John Smith 123 Main St
05674

123 Main John Smith St
05674

The AdditionalInputData field will sometimes contain the original street name or suffix if the street name was changed to obtain a match and the street name or suffix was at the end of a line. For example this address:

Precisely
4200 Parliament
Lanham MD

Validate Address would correct the spelling of the street name and add the suffix, returning "4200 Parliament PI" as the corrected street address and "Parlament" in AdditionalInputData.

Dual Addresses

A dual address is an address that contains both street and PO Box/Rural Route/Highway Contract information. Depending on the processing options you select, the portion of the dual address that is not used for address standardization may be returned in AdditionalInputData. For more information, see [About Dual Address Logic](#) on page 636.

Reports

USPS CASS 3553 Report

The USPS CASS 3553 report must be given to the USPS along with the mailing to qualify for certain discounts. The report contains information about the software you used for CASS processing, information about your name-and-address list, information about your output file, information about the mailer, and other statistics about your mailing. For detailed information about USPS Form 3553, see www.usps.com.

For more information about CASS settings see [CASS Certified Processing](#) on page 647. For instructions on how to use reports, see the *Spectrum Technology Platform Dataflow Designer Guide*.

USPS CASS Detail Report

The USPS CASS Detailed Report does not need to be given to the USPS to qualify for certain discounts. This report contains some of the same information as the 3553 report but provides much greater detail about DPV, LACS, and SuiteLink statistics.

For more information about CASS settings see [CASS Certified Processing](#) on page 647. For instructions on how to use reports, see the *Spectrum Technology Platform Dataflow Designer Guide*.

Validate Address Summary Report

The Validate Address Summary Report lists statistics about the job, such as the total number of records processed, the number of addresses validated, and more.

For instructions on how to use reports, see the *Spectrum Technology Platform Dataflow Designer Guide*.

Validate Address Global

Validate Address Global provides enhanced address standardization and validation for addresses outside the U.S. and Canada. Validate Address Global can also validate addresses in the U.S. and Canada but its strength is validation of addresses in other countries. If you process a significant number of addresses outside the U.S. and Canada, you should consider using Validate Address Global.

Validate Address Global is part of the Universal Addressing Module.

Validate Address Global performs several steps to achieve a quality address, including transliteration, parsing, validation, and formatting.

Character Set Mapping and Transliteration

Validate Address Global handles international strings and their complexities. It uses fully Unicode enabled string processing which enables the transliteration of non-roman characters into the Latin character set and mapping between different character sets.

Character set mapping and transliteration features include:

- Support for over 30 different character sets including UTF-8, ISO 8859-1, GBK, BIG5, JIS, EBCDIC
- Proper "elimination" of diacritics according to language rules
- Transliteration for various alphabets into Latin Script
- Greek (BGN/PCGN 1962, ISO 843 - 1997)
- Cyrillic (BGN/PCGN 1947, ISO 9 - 1995)
- Hebrew
- Japanese Katakana, Hiragana and Kanji
- Chinese Pinyin (Mandarin, Cantonese)
- Korean Hangul

Address Parsing, Formatting, and Standardization

Restructuring incorrectly fielded address data is a complex and difficult task especially when done for international addresses. People introduce many ambiguities as they enter address data into computer systems. Among the problems are misplaced elements (such as company or personal names in street address fields) or varying abbreviations that are not only language, but also country specific. Validate Address Global identifies address elements in address lines and assigns them to the proper fields. This is an important precursor to the actual validation. Without restructuring, "no match" situations might result.

Properly identified address elements are also important when addresses have to be truncated or shortened to fit specific field length requirements. With the proper information in the right fields, specific truncation rules can be applied.

- Parses and analyzes address lines and identifies individual address elements
- Processes over 30 different character sets
- Formats addresses according to the postal rules of the country of destination
- Standardizes address elements (such as changing AVENUE to AVE)

Global Address Validation

Address validation is the correction process where properly parsed address data is compared against reference databases supplied by postal organizations or other data providers. Validate Address Global validates individual address elements to check for correctness using sophisticated fuzzy matching technology and produces standardized and formatted output based on postal standards and user preferences. FastCompletion validation type can be used in quick address entry applications. It allows input of truncated data in several address fields and generates suggestions based on this input.

In some cases, it is not possible to fully validate an address. Here Validate Address Global has a unique deliverability assessment feature that classifies addresses according to their probable deliverability.

Input

Validate Address Global takes a standard address as input. All addresses use this format no matter what country the address is from.

Table 110: Validate Address Global Input

Field Name	Format	Description
AddressLine1 through AddressLine6	String [79]	<p>These fields contain address line data. AddressLine1 contains the first address line, AddressLine2 contains the second address line, and so forth. Note that the city, state/province, and postal code information should be placed in their respective fields, not address line fields. For example:</p> <p>AddressLine1: 17413 Blodgett Road AddressLine2: PO Box 123 City: Mount Vernon StateProvince: WA PostalCode: 97273 Country: USA</p> <p>If the input address is not already parsed into the appropriate address line and City, StateProvince, and PostalCode fields, use the UnformattedLine fields instead of the address line fields.</p>

Field Name	Format	Description
City	String [79]	City name
StateProvince	String [79]	State or province.
PostalCode	String [79]: 99999 99999-9999 A9A9A9 A9A 9A9 9999 999	The postal code for the address. In the U.S. this is the ZIP Code®.
Contact	String [79]	The name of the addressee. For example, "Mr. Jones".
Country	String [79]	The name of the country. If no value is specified in the Force country (ISO3) or Default country (ISO3) option, you must specify a country.
FirmName	String [79]	The company or firm name.
Street	String [79]	Street
Number	Building [79]	Number
Building	String [79]	Building
SubBuilding	String [79]	SubBuilding
DeliveryService	String [79]	DeliveryService

Field Name	Format	Description
UnformattedLine1 through UnformattedLine10	String [79]	<p>Use these fields if the input address is completely unparsed and you want Validate Address Global to attempt to parse the address into the appropriate fields. For example:</p> <p>UnformattedLine1: 17413 Blodgett Road UnformattedLine2: PO Box 123 UnformattedLine3: Mount Vernon WA 97273 UnformattedLine4: USA</p> <p>This address would be parsed into these output fields:</p> <p>AddressLine1: 17413 Blodgett Road AddressLine2: PO Box 123 City: Mount Vernon StateProvince: WA PostalCode: 97273 Country: USA</p> <p>Note: If you specify input in the unformatted line fields you must specify the entire address using only unformatted line fields. Do not use other fields such as City or StateProvince in combination with unformatted line fields.</p>

Address Guidelines for Japan

Address Guidelines for Japan

For information on Japanese addresses, see the Japan Post website: <http://www.post.japanpost.jp>.

A typical Japanese address looks like this:

北海道札幌市中央区大通西28丁目3番22号

The elements of this address are described in the following table.

Address Element	Field Name	Example
Prefecture	areaName1	北海道
City (Shi)	areaName2	札幌市中央区

Address Element	Field Name	Example
Municipality Subdivision (Oaza)	areaName3	大通西
City District (Chome)	areaName4	2 8 丁目
Block/lot number	mainAddress	3 番 2 2 Block and lot numbers are the most specific address elements in Japan. Japanese addresses typically do not have street names.

For multiline addresses in Kanji, the general pattern is to enter the postal code on the first line. On the second line, enter the other address elements starting from largest (prefecture) to smallest. The name of the recipient, business, or organization is entered on the third line. For example:

100-8994
東京都中央区八重洲一丁目5番3号
東京中央郵便局

For multiline addresses using Western conventions, the order of address elements is reversed. For example:

Tokyo Central Post Office
5-3, Yaesu 1-Chome
Chuo-ku, Tokyo 100-8994

Options

Input Options

Table 111: Validate Address Global Input Options

Option	Description/Valid Values
Database	Specifies the database resource containing the postal data to use for address validation. Only databases that have been defined in the Global Database Resources panel in the Management Console are available. For more information, see the <i>Spectrum Technology Platform Administration Guide</i> .

Option	Description/Valid Values
Default country (ISO3 format)	Specifies a default country to use when the input record does not contain explicit country information. Specify the country using the ISO3 country code. If you do not specify a default country each input record must have the country specified in the Country input field. For a list of ISO codes see ISO Country Codes and Module Support .
Force country (ISO3 format)	Causes address records to be always treated as originating from the country specified here, overriding the country in the address record and the default country. Specify the country using the ISO3 country code. For a list of ISO codes, see ISO Country Codes and Module Support .
Format Delimiter	<p>Enables you to use non-standard formatting for multiline addresses in input files. Acceptable values for this field include the following:</p> <ul style="list-style-type: none"> • CRLF (default) • LF • CR • SEMICOLON (2101 MASSACHUSETTS AVE NW ; WASHINGTON DC 20008) • COMMA (2101 MASSACHUSETTS AVE NW , WASHINGTON DC 20008) • TAB (2101 MASSACHUSETTS AVE NW WASHINGTON DC 20008) • PIPE (2101 MASSACHUSETTS AVE NW WASHINGTON DC 20008) • SPACE (2101 MASSACHUSETTS AVE NW WASHINGTON DC 20008) <p>Note: The same value must be selected for both the input option and output option.</p>

Output Options

Table 112: Validate Address Global Output Options

Option	Description
Maximum number of results returned	This option specifies the maximum number of candidate addresses to return. This field is disabled for batch processing; for all other processing modes the default is 1 and the maximum is 99. If you are using FastCompletion mode, you may want to enter a number greater than 1 to ensure you are provided with multiple options for completing a field.
Return input data with results	Specifies whether to include the input data in the output. If enabled, the output will contain fields that end with .Input containing the corresponding input field. For example, the output field AddressLine1.Input would contain the data specified in the input field AddressLine1.

Option	Description
State/Province	<p data-bbox="548 380 1273 407">Specifies the format for the StateProvince field. One of the following.</p> <p data-bbox="558 426 1349 485">Abbreviation Return the abbreviation for the state or province. For example, North Carolina would be returned as "NC".</p> <p data-bbox="558 504 1419 562">Country standard Return either the abbreviation or the full name depending on the format used by the country's postal authority. (Default)</p> <p data-bbox="558 581 1344 642">Extended Return the full name of the state or province, not the abbreviation. For example "North Carolina".</p>
Country format	<p data-bbox="548 730 1398 789">Specifies the language or code to use for the country name returned by Validate Address Global.</p> <ul data-bbox="548 808 1133 1507" style="list-style-type: none"> <li data-bbox="548 808 662 835">• Chinese <li data-bbox="548 842 646 869">• Danish <li data-bbox="548 875 638 903">• Dutch <li data-bbox="548 909 743 936">• English (default) <li data-bbox="548 942 646 970">• Finnish <li data-bbox="548 976 646 1003">• French <li data-bbox="548 1010 662 1037">• German <li data-bbox="548 1043 638 1071">• Greek <li data-bbox="548 1077 686 1104">• Hungarian <li data-bbox="548 1110 1133 1138">• ISO number (returns the ISO number for the country) <li data-bbox="548 1144 1109 1171">• ISO2 (returns the two-character ISO country code) <li data-bbox="548 1178 1125 1205">• ISO3 (returns the three-character ISO country code) <li data-bbox="548 1211 638 1239">• Italian <li data-bbox="548 1245 678 1272">• Japanese <li data-bbox="548 1278 646 1306">• Korean <li data-bbox="548 1312 638 1339">• Polish <li data-bbox="548 1346 695 1373">• Portuguese <li data-bbox="548 1379 662 1407">• Russian <li data-bbox="548 1413 662 1440">• Sanskrit <li data-bbox="548 1446 662 1474">• Spanish <li data-bbox="548 1480 662 1507">• Swedish

Option	Description														
Script/Alphabet	<p>Specifies the alphabet in which the output should be returned. The alphabet in which the data is returned differs from country to country. For most countries the output will be Latin I regardless of the selected preferred language.</p> <table border="0"> <tr> <td data-bbox="558 485 732 512">ASCII extended</td> <td data-bbox="870 485 1417 548">ASCII characters with expansion of special characters (for example, Ã– = OE)</td> </tr> <tr> <td data-bbox="558 564 740 592">ASCII simplified</td> <td data-bbox="870 564 1052 592">ASCII characters</td> </tr> <tr> <td data-bbox="558 611 662 638">Database</td> <td data-bbox="870 611 1417 674">(default) Latin I or ASCII characters (as per reference database standard)</td> </tr> <tr> <td data-bbox="558 690 613 718">Latin</td> <td data-bbox="870 690 1052 718">Latin I characters</td> </tr> <tr> <td data-bbox="558 737 721 764">Latin alternate</td> <td data-bbox="870 737 1336 764">Latin I characters (alternative transliteration)</td> </tr> <tr> <td data-bbox="558 783 813 810">Postal admin alternate</td> <td data-bbox="870 783 1417 846">Latin I or ASCII characters (local postal administration alternative)</td> </tr> <tr> <td data-bbox="558 863 818 890">Postal admin preferred</td> <td data-bbox="870 863 1385 926">Latin I or ASCII characters (as preferred by local postal administration)</td> </tr> </table> <p>For countries that use an alphabet other than Latin I, the returned alphabet differs from country to country. For more information, see Alphabets for Non-Latin 1 Countries on page 702.</p>	ASCII extended	ASCII characters with expansion of special characters (for example, Ã– = OE)	ASCII simplified	ASCII characters	Database	(default) Latin I or ASCII characters (as per reference database standard)	Latin	Latin I characters	Latin alternate	Latin I characters (alternative transliteration)	Postal admin alternate	Latin I or ASCII characters (local postal administration alternative)	Postal admin preferred	Latin I or ASCII characters (as preferred by local postal administration)
ASCII extended	ASCII characters with expansion of special characters (for example, Ã– = OE)														
ASCII simplified	ASCII characters														
Database	(default) Latin I or ASCII characters (as per reference database standard)														
Latin	Latin I characters														
Latin alternate	Latin I characters (alternative transliteration)														
Postal admin alternate	Latin I or ASCII characters (local postal administration alternative)														
Postal admin preferred	Latin I or ASCII characters (as preferred by local postal administration)														
Language	<p>Specifies the language in which the output should be returned. The alphabet in which the data is returned differs from country to country, but for most countries the output will be Latin, regardless of the selected preferred language.</p> <table border="0"> <tr> <td data-bbox="558 1224 662 1251">Database</td> <td data-bbox="797 1224 1398 1287">Language derived from reference data for each address. Default.</td> </tr> <tr> <td data-bbox="558 1304 646 1331">English</td> <td data-bbox="797 1304 1417 1331">English locality and state/province names output, if available.</td> </tr> </table>	Database	Language derived from reference data for each address. Default.	English	English locality and state/province names output, if available.										
Database	Language derived from reference data for each address. Default.														
English	English locality and state/province names output, if available.														
Format Delimiter	<p>Enables you to use non-standard formatting for multiline addresses in the output. Acceptable values for this field include the following:</p> <ul style="list-style-type: none"> • CRLF (default) • LF • CR • SEMICOLON (2101 MASSACHUSETTS AVE NW ; WASHINGTON DC 20008) • COMMA (2101 MASSACHUSETTS AVE NW , WASHINGTON DC 20008) • TAB (2101 MASSACHUSETTS AVE NW WASHINGTON DC 20008) • PIPE (2101 MASSACHUSETTS AVE NW WASHINGTON DC 20008) • SPACE (2101 MASSACHUSETTS AVE NW WASHINGTON DC 20008) <p>Note: The same value must be selected for both the input option and output option.</p>														

Option	Description
Casing	Specifies the casing of the output.
Native	Output will be based on the reference database standard.
Upper	Output will be in upper case for all countries.
Lower	Output will be in lower case for all countries.
Mixed	Casing determined by country-specific rules.
No change	For parse mode, returns the data the way it was entered. For validation mode, uses the casing found in the reference data and according to postal rules. Values that could not be checked against the reference data will retain their input casing.

Alphabets for Non-Latin 1 Countries

For countries that use an alphabet other than Latin I, the returned alphabet differs from country to country. The following table shows how the output is returned for specific countries. All countries that are not listed use the value specified in the **Script/Alphabet** field option.

Country	Database	Postal admin preferred	Postal admin alternate	Latin	Latin alternate	ASCII simplified	ASCII extended
RUS	Cyrillic	Cyrillic	Cyrillic	CYRILLIC_ISO	CYRILLIC_BGN	CYRILLIC_ISO + LATIN_SIMPLE	CYRILLIC_ISO + LATIN
JPN	Kanji	Kanji	Kana	JAPANESE	JAPANESE	JAPANESE + LATIN_SIMPLE	JAPANESE + LATIN
CHN	Hanzi	Hanzi	Hanzi	CHINESE_MANDARIN	CHINESE_CANTONESE	CHINESE_MANDARIN + LATIN_SIMPLE	CHINESE_MANDARIN + LATIN
HKG	Hanzi	Hanzi	Hanzi	CHINESE_CANTONESE	CHINESE_MANDARIN	CHINESE_CANTONESE + LATIN_SIMPLE	CHINESE_CANTONESE + LATIN

Country	Database	Postal admin preferred	Postal admin alternate	Latin	Latin alternate	ASCII simplified	ASCII extended
TWN	Hanzi	Hanzi	Hanzi	CHINESE_ CANTONESE	CHINESE_ MANDARIN	CHINESE_ CANTONESE + LATIN_SIMPLE	CHINESE_ CANTONESE + LATIN
GRC	Greek	Greek	Greek	GREEK_ISO	GREEK_BGN	GREEK_ISO + LATIN_SIMPLE	GREEK_ISO + LATIN
KOR	Latin	Hangul	Hanja	KOREAN	KOREAN	KOREAN + LATIN_SIMPLE	KOREAN + LATIN
ISR	Latin	Hebrew	Hebrew	HEBREW	HEBREW	HEBREW + LATIN_SIMPLE	HEBREW + LATIN
ROM	Latin-3	Latin-3	Latin-3	Latin-3	Latin-3	LATIN_SIMPLE	LATIN
POL	Latin-2	Latin-2	Latin-2	Latin-2	Latin-2	LATIN_SIMPLE	LATIN
CZE	Latin-2	Latin-2	Latin-2	Latin-2	Latin-2	LATIN_SIMPLE	LATIN
CRI	Latin-2	Latin-2	Latin-2	Latin-2	Latin-2	LATIN_SIMPLE	LATIN
HUN	Latin-2	Latin-2	Latin-2	Latin-2	Latin-2	LATIN_SIMPLE	LATIN
MDA	Latin-2	Latin-2	Latin-2	Latin-2	Latin-2	LATIN_SIMPLE	LATIN
SVK	Latin-2	Latin-2	Latin-2	Latin-2	Latin-2	LATIN_SIMPLE	LATIN
LAT	Latin-7	Latin-7	Latin-7	Latin-7	Latin-7	LATIN_SIMPLE	LATIN

Process Options**Table 113: Validate Address Global Process Options**

Option	Description
Optimization level	<p data-bbox="488 552 1425 606">Use this option to set the appropriate balance between processing speed and quality. One of the following:</p> <p data-bbox="488 627 1344 682">Narrow The parser will honor input assignment strictly, with the exception of separation of House Number from Street information.</p> <p data-bbox="488 705 1333 982">Standard The parser will separate address element more actively as follows:</p> <ul data-bbox="630 758 1252 982" style="list-style-type: none"> • Province will be separated from Locality information • PostalCode will be separated from Locality information • House Number will be separated from Street information • SubBuilding will be separated from Street information • DeliveryService will be separated from Street information • SubBuilding will be separated from Building information • Locality will be separated from PostalCode information <p data-bbox="488 1014 1417 1136">Wide Parser separation will happen similarly to Standard, but additionally up to 10 parsing candidates will be passed to validation for processing. Validation will widen its search tree and take additional reference data entries into account for matching.</p> <p data-bbox="488 1165 1398 1247">Please note that adjusting the optimization level might have no effect for countries that lack the postal reference data information required for the kind of separation described above.</p> <p data-bbox="488 1268 1403 1381">Increasing separation granularity from Narrow to Standard consumes some processing power, but the major impact on processing speed is from validation processing a larger search tree, thus increasing the number of data accesses and comparisons for the optimization level Wide, in an attempt to make the most out of the input data given.</p>

Option	Description
Processing mode	Specifies the type of processing to perform on the addresses. One of the following: <ul style="list-style-type: none"> <li data-bbox="496 426 1419 709"> <p>Batch Use this mode in batch processing environments when no human input or selection is possible. It is optimized for speed and will terminate its attempts to correct an address when ambiguous data is encountered that cannot be corrected automatically. The Batch processing mode will fall back to Parse mode when the database is missing for a specific country.</p> <p>Note: When the Process Status returns a value of I3, the attempt is considered a failure and the Status will return a value of F.</p> <li data-bbox="496 753 1419 905"> <p>Certified Use this mode in batch processing environments for Australian mail. Validate Address Global is certified by Australia Post's Address Matching Approval System (AMAS). It will standardize and validate your mail against the Postal Address File, providing postal discounts and allowing for the least amount of undeliverable pieces.</p> <li data-bbox="496 930 1419 1115"> <p>FastCompletion Use this mode if you want to use FastCompletion mode to enter truncated data in address fields and have Validate Address Global generate suggestions. For example, if you work in a call center or point-of-sale environment, you can enter just part of an address element and the FastCompletion feature will provide valid options for the complete element.</p> <li data-bbox="496 1140 1419 1415"> <p>Interactive Use this mode when working in interactive environments to generate suggestions when an address input is ambiguous. This validation type is especially useful in data entry environments when capturing data from customers or prospects. It requires the input of an almost-complete address and will attempt to validate or correct the data provided. If ambiguities are detected, this validation type will generate up to 20 suggestions that can be used for pick lists. The Interactive processing mode will fall back to Parse mode when the respective database is missing for a specific country.</p> <li data-bbox="496 1440 1419 1591"> <p>Parse Use this mode for separating address input into tokens for subsequent processing in other systems, bypassing validation. For example, you could use this mode when address data of already high quality simply needs to be tokenized quickly for export to an external system or for use by a downstream stage.</p>

Option	Description
Matching scope	<p>Specifies how closely an address must match the reference data in order for the address to be validated. One of the following:</p> <p>Note: These settings may not have an effect for countries lacking the necessary level of detail in the postal reference data.</p> <p>All levels All address elements must match.</p> <p>Delivery point level Validate Global Address must achieve a match on StateProvince, PostalCode, City/Locality/Suburb, street, house number, and sub building.</p> <p>Street level Validate Global Address must achieve a match on StateProvince, PostalCode, City/Locality/Suburb, and street.</p> <p>Locality level Validate Global Address must achieve a match on StateProvince, PostalCode, and City/Locality/Suburb.</p>

Output

Address Data

Table 114: Parsed Address Elements

Field Name	Description
AddressBlock1-9	<p>The AddressBlock output fields contain a formatted version of the standardized or normalized address as it would be printed on a physical mailpiece. Validate Address Global formats the address into address blocks using postal authority standards. Each line of the address is returned in a separate address block field. There can be up to nine address block output fields: AddressBlock1 through AddressBlock9. For example, this input address:</p> <p>AddressLine1: 4200 Parliament Place AddressLine2: Suite 600 City: Lanham StateProvince: MD PostalCode: 20706</p> <p>Results in this address block output:</p> <p>AddressBlock1: 4200 PARLIAMENT PL STE 600 AddressBlock2: LANHAM MD 20706-1882</p>

Field Name	Description
AddressLine1-6	<p>If the address was validated, the address line fields contain the validated and standardized address lines. If the address could not be validated, the address line fields contain the input address without any changes. Note that the last line of the address is contained in the LastLine field. For example:</p> <p>AddressLine1: 4200 PARLIAMENT PL STE 600 LastLine: LANHAM MD 20706-1882</p>
AdministrativeDistrict	An area smaller than a state/province but larger than a city.
ApartmentLabel	The flat or unit type (such as STE or APT), for example: 123 E Main St Apt 3
ApartmentNumber	The flat or unit number, for example: 123 E Main St Apt 3
BlockName	An estate or block name.
BuildingName	The name of a building, for example Sears Tower.
City	The name of the town or city. For example, Vancouver , BC.
City.AddInfo	Additional information about the city.
City.SortingCode	A code used by the postal authority to speed up delivery in certain countries for large localities, for example Prague or Dublin.
Contact	The name of the addressee. For example, Mr. Jones .
Country	The country in the language or code specified in the Country format option.
County	Dependent state or province information that further subdivides a state or province. An example would be a U.S. county.
FirmName	The name of a company.
Floor	Information that further subdivides a building, for example, the suite or apartment number. For example: 123 E Main St Apt 3, 4th Floor

Field Name	Description
HouseNumber	The house number 1, for example: 298A-1B New South Head Rd
LastLine	Complete last address line (city, state/province, and postal code).
LeadingDirectional	Street directional that precedes the street name. For example, the N in 138 N Main Street.
Locality	Dependent place name that further subdivides a Locality. Examples are colonias in Mexico, Urbanisaciones in Spain.
POBox	Post Box descriptor (POBox, Postfach, Case Postale etc.) and number.
PostalCode	The postal code for the address. The format of the postcode varies by country.
PostalCode.AddOn	The second part of a postcode. For example, for Canadian addresses this will be the LDU. For U.S. addresses this is the ZIP + 4 add on. This field is not used by most countries.
PostalCode.Base	The base portion of the postcode.
Room	A room number in a building.
SecondaryStreet	The name of a secondary street or rural route.
StateProvince	The name of the state or province.
StreetName	The name of street where property is located, for example: 123 E Main St Apt 3
StreetSuffix	The street suffix, for example: 123 E Main St Apt 3
SubBuilding	A portion of a building, such as a suite. For example, Suite 102.
Suburb	Dependent place name that further subdivides a Locality. An example would be Mahalle in Turkey.
Territory	The name of a territory. Territories are larger than a state/province.

Field Name	Description
TrailingDirectional	The trailing directional, for example: 123 Pennsylvania Ave NW

Original Input Data

This option outputs the original input data in <FieldName>.Input fields.

Table 115: Original Input Data

Field Name	Format	Description
AddressLine1.Input	String [79]	First address line
AddressLine2.Input	String [79]	Second address line
AddressLine3.Input	String [79]	Third address line
AddressLine4.Input	String [79]	Fourth address line
AddressLine5.Input	String [79]	Fifth address line
AddressLine6.Input	String [79]	Sixth address line
City.Input	String [79]	City name
StateProvince.Input	String [79]	State or province

Field Name	Format	Description
PostalCode.Input	String [79]	The postal code for the address. In the U.S. this is the ZIP Code. One of these formats: 99999 99999-9999 A9A9A9 A9A 9A9 9999 999
Contact.Input	String [79]	The name of the addressee. For example, "Mr. Jones".
Country.Input	String [79]	Specify the country using the format you chose for input country format (English name, ISO code, or UPU code). For a list of valid values, see ISO Country Codes and Module Support .
FirmName.Input	String [79]	The company or firm name.
Street.Input	String [79]	Street
Number.Input	Building [79]	Number
Building.Input	String [79]	Building
SubBuilding.Input	String [79]	SubBuilding
DeliveryService.Input	String [79]	DeliveryService

Result Codes

These output fields contain information about the result of the validation processing.

Table 116: Result Codes

Field Name	Result Code
AddressType	<p>For United States and Canada addresses only, the AddressType field indicates the type of address. One of the following:</p> <p>F The address was validated/corrected to the firm name.</p> <p>B The address was validated/corrected to the building name.</p> <p>G The address is a general delivery address.</p> <p>H The address was validated/corrected to the high-rise default.</p> <p>L The address is a large volume receiver.</p> <p>M The address is a military address.</p> <p>P The address was validated/corrected to PO box.</p> <p>R The address was validated/corrected to a rural route.</p> <p>S The address was validated/corrected to a street address.</p> <p>U The address could not be validated/corrected so the type is unknown.</p>
Confidence	<p>The level of confidence assigned to the address being returned. Range is from zero (0) to 100; zero indicates failure, 100 indicates a very high level of confidence that the match results are correct.</p>
CountOverflow	<p>Indicates whether the number of candidate addresses exceeds the number returned. One of the following:</p> <p>Yes Yes, there are additional candidate addresses. To obtain the additional candidates, increase the Maximum number of results returned value.</p> <p>No No, there are no additional candidates.</p>
ElementInputStatus	<p>ElementInputStatus provides per element information on the matching of input elements to reference data. The values in this field vary depending on whether you are using batch mode or parse mode. For information about the value in this field, see Interpreting ElementInputStatus, ElementResultStatus, and ElementRelevance on page 716.</p>
ElementRelevance	<p>Indicates which address elements are actually relevant from the local postal authority's point of view. For information about the value in this field, see Interpreting ElementInputStatus, ElementResultStatus, and ElementRelevance on page 716 .</p>

Field Name	Result Code												
ElementResultStatus	ElementResultStatus categorizes the result in more detail than the ProcessStatus field by indicating if and how the output fields have been changed from the input fields. For information about the value in this field, see Interpreting ElementInputStatus, ElementResultStatus, and ElementRelevance on page 716.												
MailabilityScore	<p>An estimate of how likely it is that mail sent to the address would be successful delivered. One of the following:</p> <table border="0"> <tr> <td>5</td> <td>Completely confident of deliverability</td> </tr> <tr> <td>4</td> <td>Almost certainly deliverable</td> </tr> <tr> <td>3</td> <td>Should be deliverable</td> </tr> <tr> <td>2</td> <td>Fair chance</td> </tr> <tr> <td>1</td> <td>Risky</td> </tr> <tr> <td>0</td> <td>No chance</td> </tr> </table>	5	Completely confident of deliverability	4	Almost certainly deliverable	3	Should be deliverable	2	Fair chance	1	Risky	0	No chance
5	Completely confident of deliverability												
4	Almost certainly deliverable												
3	Should be deliverable												
2	Fair chance												
1	Risky												
0	No chance												
ModeUsed	Indicates the processing mode used. The processing mode is specified in the Processing Mode option. For a description of the modes, see Process Options on page 704.												
MultimatchCount	If the address was matched to multiple candidate addresses in the reference data, this field contains the number of candidate matches found.												

Field Name	Result Code
------------	-------------

ProcessStatus	
---------------	--

Field Name

Result Code

Provides a general description of the output quality. For a more detailed description of the output quality, see the ElementResultStatus field.

One of the following:

V4	Verified. The input data is correct. All elements were checked and input matched perfectly.
V3	Verified. The input data is correct on input but some or all elements were standardized or the input contains outdated names or exonyms.
V2	Verified. The input data is correct but some elements could not be verified because of incomplete reference data.
V1	Verified. The input data is correct but the user standardization has deteriorated deliverability (wrong element user standardization - for example, postcode length chosen is too short). Not set by validation.
C4	Corrected. All elements have been checked.
C3	Corrected, but some elements could not be checked.
C2	Corrected, but delivery status unclear (lack of reference data).
C1	Corrected, but delivery status unclear because user standardization was wrong. Not set by validation.
I4	Data could not be corrected completely, but is very likely to be deliverable. Single match (for example, HNO is wrong but only 1 HNO is found in reference data).
I3	Data could not be corrected completely, but is very likely to be deliverable. Multiple matches (for example, HNO is wrong but more than 1 HNO is found in reference data).
I2	Data could not be corrected, but there is a slim chance that the address is deliverable.
I1	Data could not be corrected and is unlikely to be delivered.
RA	Country recognized from the Force country Setting
R9	Country recognized from DefaultCountryISO3 Setting
R8	Country recognized from name without errors
R7	Country recognized from name with errors
R6	Country recognized from territory
R5	Country recognized from province
R4	Country recognized from major town
R3	Country recognized from format
R2	Country recognized from script
R1	Country not recognized - multiple matches

Field Name	Result Code
	R0 Country not recognized
	S4 Parsed perfectly
	S3 Parsed with multiple results
	S2 Parsed with errors. Elements change position.
	S1 Parse Error. Input Format Mismatch.
	N1 Validation Error: No validation performed because country was not recognized.
	N2 Validation Error: No validation performed because required reference database is not available.
	N3 Validation Error: No validation performed because country could not be unlocked.
	N4 Validation Error: No validation performed because reference database is corrupt or in wrong format.
	N5 Validation Error: No validation performed because reference database is too old.
	N6 Validation Error: No validation performed because input data was insufficient.
	Q3 FastCompletion Status: Suggestions are available - complete address.
	Q2 FastCompletion Status: Suggested address is complete but combined with elements from the input (added or deleted).
	Q1 FastCompletion Status: Suggested address is not complete (enter more information).
	Q0 FastCompletion Status: Insufficient information provided to generate suggestions.
Status	Reports the success or failure of the processing attempt.
	null Success
	F Failure
Status.Code	The reason for the failure, if there was one.
Status.Description	A description of the reason for the failure, if there was one.

Interpreting ElementInputStatus, ElementResultStatus, and ElementRelevance

The ElementInputStatus, ElementResultStatus, and ElementRelevance output fields contain a series of digits that describe the outcome of the validation operation in detail. ElementInputStatus contains some information for parsing operations.

This is what an ElementInputStatus value looks like:

44606040600000000060

This is what an ElementResultStatus value looks like:

88F0F870F00000000040

This is what an ElementRelevance value looks like:

11101010100000000000

To understand the values in these fields you need to know which element each position represents, and the meaning of the values in each position. For example, the first digit indicates the result from the PostalCode.Base output field. The position meanings are listed below.

- Position 1—PostalCode.Base
- Position 2—PostalCode.AddOn
- Position 3—City
- Position 4—Locality and Suburb
- Position 5—StateProvince
- Position 6—County
- Position 7—StreetName
- Position 8—SecondaryStreet
- Position 9—HouseNumber
- Position 10—Number level 1
- Position 11—POBox
- Position 12—Delivery service level 1
- Position 13—Building level 0
- Position 14—BuildingName
- Position 15—Sub building level 0
- Position 16—Floor and Room
- Position 17—FirmName
- Position 18—Organization level 1
- Position 19—Country
- Position 20—Territory

For ElementInputStatus, the possible values for validation are:

- 0—Empty
- 1—Not found
- 2—Not checked (no reference data)

- 3—Wrong - Set by validation only: The reference database suggests that either Number or DeliveryService is out of valid number range. Input is copied, not corrected for batch mode, for interactive mode and FastCompletion suggestions are provided.
- 4—Matched with errors in this element
- 5—Matched with changes (inserts and deletes) For example:
 - Parsing: Splitting of house number for "MainSt 1"
 - Validation: Replacing input that is an exonym or dropping superfluous fielded input that is invalid according to the country reference database
- 6—Matched without errors

For ElementInputStatus, the possible values for parsing are:

- 0—Empty
- 1—Element had to be relocated
- 2—Matched but needed to be normalized
- 3—Matched

For ElementRelevance, the possible values for parsing are:

- 0—Empty
- 1—Element had to be relocated
- 2—Matched but needed to be normalized
- 3—Matched

For ElementResultStatus, the possible values are (for all address elements apart from country):

- 0—Empty
- 1—Not validated and not changed. Original is copied.
- 2—Not validated but standardized.
- 3—Validated but not changed due to invalid input, database suggests that number is out of valid ranges. Input is copied, not corrected - this status value is only set in batch mode.
- 4—Validated but not changed due to lack of reference data.
- 5—Validated but not changed due to multiple matches. Only set in batch mode, otherwise multiple suggestions that replace the input are marked as corrected (status value 7).
- 6—Validated and changed by eliminating the input value
- 7—Validated and changed due to correction based on reference data
- 8—Validated and changed by adding value based on reference data
- 9—Validated, not changed, but delivery status not clear (for example, DPV value wrong; given number ranges that only partially match reference data).
- C—Validated, verified but changed due to outdated name
- D—Validated, verified but changed from exonym to official name
- E—Validated, verified but changed due to standardization based on casing or language. Validation only sets this status if input fully matches a language alternative.
- F—Validated, verified and not changed due to perfect match

For Country (position 19 & 20), the following values are possible:

- 0—Empty
- 1—Country not recognized
- 4—Country recognized from DefaultCountryISO3 setting
- 5—Country not recognized - multiple matches
- 6—Country recognized from script
- 7—Country recognized from format
- 8—Country recognized from major town
- 9—Country recognized from province
- C—Country recognized from territory
- D—Country recognized from name with errors
- E—Country recognized from name without errors
- F—Country recognized from ForceCountryISO3 setting

Reports

Validate Address Global Summary Report

The Validate Address Global Summary Report lists summary statistics about the job, such as the total number of records processed, the number of addresses validated, and more. For instructions on how to use reports, see the *Spectrum Technology Platform Dataflow Designer's Guide*.

Job Summary

This section contains summary information about the job.

- **Started**—The date and time that the job started.
- **Finished**—The date and time that the job ended.
- **Processing time**—The duration of the job.
- **Total Records**—The total number of records presented to Validate Address Global for processing. This may be different from the number of input records for the job depending on how the job is designed.
- **Processed Records**—The number of addresses that were successfully processed by Validate Address Global. This is the total number of records less records not processed.
- **Default country**—The default country specified in the **Default country (ISO3 format)** option.
- **Casing**—The casing selected in the **Casing** option.
- **Script/Alphabet**—The script specified in the **Script/Alphabet** option.
- **Countries**—The number of countries represented in the input addresses.

Status Summary

This section lists the validation and correction results.

- **Validated**—Addresses that were correct on input.

- **Corrected**—Addresses that were corrected by Validate Address Global.
- **Good deliverability**—Addresses that could not be corrected but that are very likely to be deliverable.
- **Fair deliverability**—Addresses that could not be corrected but have a fair chance that the address is deliverable.
- **Poor deliverability**—Addresses that could not be corrected and are unlikely to be deliverable.
- **Parsed**—Addresses that were successfully parsed.
- **Failed**—Addresses that could not be verified, corrected, or parsed.

Validate Address Global Detail Report

The Validate Address Detail Report shows the results of validation/correction/parsing for each country. For instructions on how to use reports, see the *Spectrum Technology Platform Dataflow Designer's Guide*.

Status Details

This section lists the validation and correction results for each country.

- **V (Validated)**—Addresses that were correct on input.
- **C (Corrected)**—Addresses that were corrected by Validate Address Global.
- **I4 (Good deliverability)**—Addresses that could not be corrected but that are very likely to be deliverable.
- **I3 (Fair deliverability)**—Addresses that could not be corrected but have a fair chance that the address is deliverable.
- **I2 (Poor deliverability)**—Addresses that could not be corrected and are unlikely to be deliverable.
- **S (Parsed)**—Addresses that were successfully parsed.
- **F (Failed)**—Addresses that could not be verified, corrected, or parsed.

Validate Address Loqate

Validate Address Loqate standardizes and validates addresses using postal authority address data. Validate Address Loqate can correct information and format the address using the format preferred by the applicable postal authority. It also adds missing postal information, such as postal codes, city names, state/province names, and so on.

Validate Address Loqate also returns result indicators about validation attempts, such as whether or not Validate Address Loqate validated the address, the level of confidence in the returned address, the reason for failure if the address could not be validated, and more.

During address matching and standardization, Validate Address Loqate separates address lines into components and compares them to the contents of the Spectrum Universal Address databases. If a match is found, the input address is *standardized* to the database information. If no database match is found, Validate Address Loqate optionally *formats* the input addresses. The formatting process attempts to structure the address lines according to the conventions of the appropriate postal authority.

Validate Address Loqate is part of Spectrum Universal Address.

Input

Validate Address Loqate takes an address as input. All addresses use this format regardless of the address's country.

Table 117: Input Format

Field Name	Format	Description
AddressLine1	String	The first address line.
AddressLine2	String	The second address line.
AddressLine3	String	The third address line.
AddressLine4	String	The fourth address line.
City	String	The city name.
Country	String	<p>The country code or name, in any of the following formats:</p> <ul style="list-style-type: none"> • Two-character ISO 3166-1 Alpha 2 country code • Three-character ISO 3166-1 Alpha 3 country code • English country name <p>See ISO Country Codes and Module Support for a list of ISO codes. See ISO Country Code and Precision Level for a list of ISO codes.</p>
FirmName	String	The company or firm name.
PostalCode	String	<p>The postal code for the address in one of these formats:</p> <p>99999 99999-9999 A9A9A9 A9A 9A9 9999 999</p>

Field Name	Format	Description
StateProvince	String	The state or province.

Options

The following table lists the options that control the type of information returned by Validate Address Loqate.

Table 118: Output Data Options

Option	Description
Database	Specifies which database you want to use for validating international addresses. To specify a database for international address validation, select a database in the Database drop-down list.
Include a standard address	<p>Returns 1 to 4 lines of address data plus city, state, postal code, and firm name. Each address line represents an actual line of the address as it would appear on an envelope. For more information, see Output on page 728.</p> <p>If Validate Address Loqate could validate the address, the address lines contain the standardized address. When addresses are standardized, punctuation is removed, directionals are abbreviated, street suffixes are abbreviated, and address elements are corrected.</p> <p>If Validate Address Loqate could not validate the address, the address lines contain the address as it appeared in the input ("pass through" data). Non-validated addresses are always included as pass through data in the address line fields even if you uncheck this option.</p>
Include matched address elements	Each part of the address, such as house number, street name, street suffix, directionals, and so on is returned in a separate field. For more information, see Parsed Address Elements Output on page 729. Note that if you select this option and also select Return normalized data when no match is found , the address elements will contain the input address for addresses that could not be validated.

Option	Description
Include standardized input address elements	<p>This option returns the input address in parsed form regardless of whether or not Validate Address Loqate is able to validate the address. Each part of the input address, such as house number, street name, street suffix, directionals, and so on is returned in a separate field.</p> <p>Selecting this option differs from selecting the combination of Include matched address elements/Return normalized data when no match is found in that Return standardized input address elements returns all input address in parsed form, not just input that could not be validated. For more information, see Parsed Input on page 732.</p>
Return geocoded address fields	<p>Specifies whether to perform geocoding during processing. Geocoding output provides the latitude and longitude for each input address, as well as the level of accuracy of the match and the likely maximum distance between the geocode and the actual physical location of the address.</p>
Include result codes for individual fields	<p>Specifies whether to include field-level result indicators. Field-level result indicators describe how Validate Address Loqate handled each address element. Field-level result indicators are returned in the qualifier "Result". For example, the field-level result indicator for HouseNumber is contained in HouseNumber.Result. For a complete listing of result indicator output fields, see Result Indicators on page 736.</p>
Return normalized data when no match is found	<p>Specifies whether to return a formatted address when an address cannot be validated. The address is formatted using the preferred address format for the address's country. If this option is not selected, the output address fields are blank when Validate Address Loqate cannot validate the address.</p> <p>Formatted addresses are returned using the format specified by the Include a standard address, Include address line elements, and Include postal information check boxes. Note that if you select Include address line elements, the parsed address elements will contain the parsed, validated address for addresses that could be validated. If the address could not be validated the parsed address elements will contain the input address in parsed form. If you always want the output to contain the input address in parsed form, regardless of whether or not Validate Address Loqate could validate the address, select Include standardized input address elements.</p> <p>If you check this option, you must select Include a standard address and/or Include address line elements.</p>

Option	Description
Return address data blocks	<p>Specifies whether to return a formatted version of the address as it would be printed on a physical mailpiece. Each line of the address is returned in a separate address block field. There can be up to nine address block output fields: AddressBlock1 through AddressBlock9.</p> <p>For example, this input address:</p> <p>AddressLine1: 4200 Parliament Place AddressLine2: Suite 600 City: Lanham StateProvince: MD PostalCode: 20706</p> <p>Results in this address block output:</p> <p>AddressBlock1: 4200 PARLIAMENT PL STE 600 AddressBlock2: LANHAM MD 20706-1882 AddressBlock3: UNITED STATES OF AMERICA</p> <p>Validate Address Loqate formats the address into address blocks using postal authority standards. The country name is returned using the Universal Postal Union country name. Note that the option Country format does not affect the country name in the address block, it only affects the name returned in the Country output field.</p>
Format data using AMAS conventions	<p>Specifies that output address data is to be formatted using Address Matching Approval System (AMAS) conventions.</p> <p>This option causes Validate Address Loqate to use AMAS rules when standardizing an address. AMAS is an Australia Post program for enforcing addressing standards. For more information on the AMAS formatting conventions, refer to the Address Matching Approval System (AMAS) Handbook.</p> <p>This option modifies the output data as follows.</p> <ul style="list-style-type: none"> • Numeric fields are padded with zeros. This affects the following output fields: HouseNumber, HouseNumber2, PostalDeliveryNumber, and DPID. For example, if the input address is 298 New South Head Rd Double Bay NSW 2028, then the format of the HouseNumber field is changed from 298 to 00298. • If a match is not made, then all digits in the DPID field will be zero. For example, 00000000. • If a match is not made, then all return fields (parsed address elements) will be blank, except numeric fields which will contain all zeros. • The CCD field is not output. <p>Note: When this option is selected, results will be returned with AMAS formatting regardless of selections made in the Acceptance level and Minimum match score fields.</p>

Option	Description
Casing	<p>Specifies the casing of the output data. One of the following:</p> <p>Mixed Returns the output in mixed case (default). For example: 123 Main St Mytown FL 12345</p> <p>Upper Returns the output in upper case. For example: 123 MAIN ST MYTOWN FL 12345</p>
Default country	<p>Specifies the default country. You should specify the country where most of your addresses reside. For example, if most of the addresses you process are in Germany, specify Germany. Validate Address Loqate uses the country you specify to attempt validation when it cannot determine the country from the StateProvince, PostalCode, and Country address fields.</p>
Country format	<p>Specifies the format to use for the country name returned in the Country output field. For example, if you select English, the country name "Deutschland" would be returned as "Germany".</p> <p>English Names Use English country names (default).</p> <p>ISO Codes Use two-letter ISO abbreviation for the countries instead of country names.</p> <p>UPU Codes Use Universal Postal Union abbreviation for the countries instead of country names.</p>
Script/Alphabet	<p>Specifies the alphabet or script in which the output should be returned. This option is bi-directional and generally takes place from Native to Latin and Latin to Native.</p> <p>Input Script Do not perform transliteration and provide output in the same script as the input (default).</p> <p>Native Output in the native script for the selected country wherever possible.</p> <p>Latin (English) Use English values.</p>

Option	Description
Acceptance level	<p>Specifies the minimum verification level a record must reach to be considered successfully processed. The value in this field corresponds to the second character of the Address Verification Code, which is called "Post-Processed Verification Match Level":</p> <ul style="list-style-type: none"> • 5—Delivery point (building or post box). The record will be passed or will have high confidence if ApartmentNumber, HouseNumber, Street, City, and StateProvince supplied in the input record match to the Loqate reference dataset. Will have moderate confidence if ApartmentNumber is correct but other remaining fields are incorrect, but in this case the Loqate engine should be able to identify the ApartmentNumber as ApartmentNumber is at a more granular level. It will have zero confidence if ApartmentNumber and other fields are unable to be parsed by the Loqate engine. • 4—Premise or building. The record will be passed or will have high confidence if House Number, Street, City, and StateProvince supplied in the input record match the Loqate reference dataset. Will have moderate confidence if HouseNumber is correct but the other fields are not; however, in this case the Loqate engine should be able to identify the HouseNumber because HouseNumber is at a more granular level. It will have zero confidence if the HouseNumber and other fields are unable to be parsed by the Loqate engine. • 3—Thoroughfare, road, or street. The record will be passed or will have high confidence if Street, City, and StateProvince supplied in the input record match the Loqate reference dataset. Will have moderate confidence if City is correct but StateProvince is not; however, in this case the Loqate engine should be able to identify the StateProvince as City itself is the part of StateProvince. It will have zero confidence if City or both fields (City and State Province) are unable to be parsed by the Loqate engine. • 2—Locality (city or town). The record will be passed or will have high confidence if both City and StateProvince supplied in the input record match the Loqate reference dataset. Will have moderate confidence if City is correct but StateProvince is not; however, in this case the Loqate Engine should be able to identify the StateProvince as City itself is the part of StateProvince. It will have zero confidence if City or both fields (City and StateProvince) are unable to be parsed by the Loqate engine. • 1—Administrative area (state or region). The record will be passed or will have high confidence if the StateProvince supplied in the input record matches the Loqate reference dataset. • 0—None. This is equivalent to loosest match option.

Option	Description
Duplicate handling	<p>Enables the duplicate handling mask and specifies how duplicate records are processed and removed. Select one or more of the following options:</p> <p>Single Selected by default. Preprocess the input and remove duplicates that occur in a single field.</p> <p>Multi Selected by default. Pre-process the input and remove duplicates across all fields.</p> <p>Non-standard Pre-process the input and remove duplicates in fields that are not standard address fields.</p> <p>Output Selected by default. Post-process the output from verification and remove duplicates from non-verified fields.</p>
Minimum match score	<p>Specifies a numeric value between 0 and 100 that indicates the degree to which Validate Address Loqate will change an address in order to obtain a match in the Loqate reference database. The lower the number, the greater amount of change is allowed. A value of 100 means that after parsing the input address is nearly identical to the validated address. A value of 0 means that the parsed input address may be completely changed in order to obtain a validated address.</p>
Return multiple addresses	<p>Specifies whether or not to return multiple address for those input addresses that have more than one possible match.</p> <p>For more information, see Returning Multiple Matches on page 726.</p>
Fail Multiple Matches	<p>Fails multiple addresses for those input addresses that have more than one possible match.</p>

Returning Multiple Matches

If Validate Address Loqate finds multiple address in the postal database that are possible matches for the input address, you can have Validate Address Loqate return the possible matches. For example, the following address matches multiple addresses in the U.S. postal database:

PO BOX 1 New York, NY

Options

To return multiple matches, use the options described in the following table.

Table 119: Multiple Match Option

Option Name	Description/Valid Values
Return multiple addresses	Indicates whether or not to return multiple address for those input addresses that have more than one possible match.
Maximum results	Next to the Return multiple addresses check box, enter a number between 1 and 10 that indicates the maximum number of addresses to return. The default value is 1. Note: The difference between unchecking Return multiple addresses and checking Return multiple addresses and specifying a maximum number of results of 1 is that a multiple match will return a failure if Return multiple addresses is unchecked, whereas a multiple match will return one record if Return multiple addresses is checked and the maximum number of results is 1.
Include result codes for individual fields	To identify which output addresses are candidate addresses, you must check Include result codes for individual fields on the Output Data tab. When you do this, records that are candidate addresses will have one or more "M" values in the field-level result indicators.

Output

When you choose to return multiple matches, the addresses are returned in the address format you specify. For information on specifying address format, see [Options](#) on page 721. To identify which records are the candidate addresses, look for multiple "M" values in the field-level result indicators. For more information, see [Result Indicators](#) on page 736.

Match Score Threshold Options

There are two options for setting match score thresholds.

Note: These options are not available in the Validate Address Loqate user interface; they are located in the following file:

```
SpectrumDirectory/server/modules/loqate/env.properties
```

The **MatchScoreAbsoluteThreshold** option is used to specify the minimum match score a record must reach to be considered a candidate for matching. The default value is 60, and the maximum value is 100.

The **MatchScoreThresholdFactor** is a value that represents a factor of the highest matching result. This value is used as a cutoff for considering result candidates. The higher the value of the factor, the higher the chance of getting a good verification result. The default value is 95 and the maximum value is 100.

Output

The output from Validate Address Loqate contains various information depending on the output categories you select.

Standard Address Output

Standard address output consists of four lines of the address which correspond to how the address would appear on an address label. City, state/province, postal code, and other data is also included in standard address output. Validate Address Loqate returns standard address output for validated addresses if you select the **Include a standard address** check box. Standard address fields are always returned for addresses that could not be validated regardless of whether or not you select the **Include a standard address** check box. For non-validated addresses, the standard address output fields contain the address as it appeared in the input ("pass through" data). If you want Validate Address Loqate to standardize address according to postal authority standards when validation fails, select the **Include normalized data when no match is found** check box.

Table 120: Standard Address Output

Field Name	Description
AdditionalInputData	Input data that could not be matched to a particular address component. For more information, see About Additional Input Data .
AddressLine1-4	If the address was validated, the first line of the validated and standardized address. If the address could not be validated, the first line of the input address without any changes. There can be up to four address block output fields: AddressLine1 through AddressLine4.
City	The validated city name.
Country	The country in the format determined by what you selected in Country format: <ul style="list-style-type: none"> • ISO Code • UPU Code • English

Field Name	Description
FirmName	The validated firm or company name.
PostalCode	The validated ZIP Code™ or postal code.
PostalCode.AddOn	The 4-digit add-on part of the ZIP Code™. For example, in the ZIP Code™ 60655-1844, 1844 is the 4-digit add-on.
PostalCode.Base	The 5-digit ZIP Code™; for example 20706.
StateProvince	The validated state/province or its abbreviated value.

Parsed Address Elements Output

Output addresses are formatted in the parsed address format if you select the **Include matched address elements** check box. If you want Validate Address Loqate to return formatted data in the Parsed Address format when validation fails (that is, a normalized address), select the **Return normalized data when no match is found** check box.

Note: If you want Validate Address Loqate to always return parsed input data regardless of whether or not validation is successful, select **Include standardized input address elements**. For more information, see [Parsed Input](#) on page 732.

Table 121: Parsed Address Output

Field Name	Description
AddressBlock1-9	<p>The AddressBlock output fields contain a formatted version of the standardized or normalized address as it would be printed on a physical mailpiece. Validate Address Global formats the address into address blocks using postal authority standards. Each line of the address is returned in a separate address block field. There can be up to nine address block output fields: AddressBlock1 through AddressBlock9. For example, this input address:</p> <p>AddressLine1: 4200 Parliament Place AddressLine2: Suite 600 City: Lanham StateProvince: MD PostalCode: 20706</p> <p>Results in this address block output:</p> <p>AddressBlock1: 4200 PARLIAMENT PL STE 600 AddressBlock2: LANHAM MD 20706-1882</p>
ApartmentLabel	Apartment designator (such as STE or APT), for example: 123 E Main St APT 3
ApartmentNumber	Apartment number, for example: 123 E Main St APT 3
ApartmentNumber2	<p>Secondary apartment number, for example: 123 E Main St APT 3, 4th Floor</p> <p>Note: In this release, this field will always be blank.</p>
Building	Descriptive name identifying an individual location.
City	Validated city name

Field Name	Description
Country	Country. Format is determined by what you selected in Country format : <ul style="list-style-type: none"> • ISO Code • UPU Code • English
County*	The smallest geographic data element within a country, for instance, USA County
FirmName	The validated firm or company name
HouseNumber	House number, for example: 123 E Main St Apt 3
LeadingDirectional	Leading directional, for example: 123 E Main St Apt 3
POBox	Post office box number. If the address is a rural route address, the rural route box number will appear here.
PostalCode	Validated postal code. For U.S. addresses, this is the ZIP Code.
Principality *	The largest geographic data element within a country
StateProvince	Validated state or province name
StreetAlias	Alternate street name; typically applies only to a specific range of addresses on the street. If you do not allow street aliases in the output then the street's "base" name will appear in the output regardless of whether or not there is an alias for the street. for example: 123 E Main St Apt 3
StreetName	Street name, for example: 123 E Main St Apt 3

Field Name	Description
StreetSuffix	Street suffix, for example: 123 E Main St Apt 3
Subcity*	A smaller population center data element, dependent on the contents of the Locality field. For instance, Turkish Neighbourhood .
Substreet*	The dependent street or block data element within a country. For instance, UK Dependent Street .
TrailingDirectional	Trailing directional, for example: 123 Pennsylvania Ave NW

*This is a subfield and may not contain data.

Parsed Input

The output can include the input address in parsed form. This type of output is referred to as "parsed input." Parsed input fields contain the address data that was used as input regardless of whether or not Validate Address validated the address. Parsed input is different from the "parsed address elements" output in that parsed address elements contain the validated address if the address could be validated, and, optionally, the input address if the address could not be validated. Parsed input always contains the input address regardless of whether or not Validate Address validated the address.

To include parsed input fields in the output, select the **Return parsed input data** check box.

Table 122: Parsed Input

Field Name	Description
ApartmentLabel.Input	Apartment designator (such as STE or APT), for example: 123 E Main St APT 3
ApartmentNumber.Input	Apartment number, for example: 123 E Main St APT 3
City.Input	Validated city name

Field Name	Description
Country.Input	Country. Format is determined by what you selected in Country format: <ul style="list-style-type: none"> • ISO Code • UPU Code • English
County.Input*	The smallest geographic data element within a country, for instance, USA County
FirmName.Input	The validated firm or company name
HouseNumber.Input	House number, for example: 123 E Main St Apt 3
LeadingDirectional.Input	Leading directional, for example: 123 E Main St Apt 3
POBox.Input	Post office box number. If the address is a rural route address, the rural route box number will appear here.
PostalCode.Input	Validated postal code. For U.S. addresses, this is the ZIP Code.
Principality.Input *	The largest geographic data element within a country
StateProvince.Input	Validated state or province name
StreetAlias.Input	Alternate street name; typically applies only to a specific range of addresses on the street. If you do not allow street aliases in the output then the street's "base" name will appear in the output regardless of whether or not there is an alias for the street. The base name is the name that applies to the entire street. For example: If StreetName is "N MAIN ST" the StreetAlias field would contain "MAIN" and the thoroughfare type, "ST", would be returned in the StreetSuffix field.

Field Name	Description
StreetName.Input	Street name, for example: 123 E Main St Apt 3
StreetSuffix.Input	Street suffix, for example: 123 E Main St Apt 3
Subcity.Input*	A smaller population center data element, dependent on the contents of the Locality field. For instance, Turkish Neighbourhood .
Substreet.Input*	The dependent street or block data element within a country. For instance, UK Dependent Street .
TrailingDirectional.Input	Trailing directional, for example: 123 Pennsylvania Ave NW

*This is a subfield and may not contain data.

Geocode Output

Validate Address Loqate returns the latitude/longitude, geocoding match code, dependent and double dependent localities, dependent thoroughfare, subadministrative and superadministrative areas, and the search distance as output. Match codes describe how well the geocoder matched the input address to a known address; they also describe the overall status of a match attempt. Search distance codes represent how close the geocode is to the actual physical location of an address.

Table 123: Geocode Address Output

Field Name	Description
Geocode.MatchCode	<p>This two-byte code reflects the status and level of geocode matching for an address. The first byte represents the geocoding status and is one of the following:</p> <p>A Multiple candidate geocodes were found to match the input address, and an average of these was returned</p> <p>I A geocode was able to be interpolated from the input addresses location in a range</p> <p>P A single geocode was found matching the input address</p> <p>U A geocode was not able to be generated for the input address</p> <p>The second byte represents the level of geocoding matching and is one of the following:</p> <p>5 Delivery point (post box or subbuilding)</p> <p>4 Premise or building</p> <p>3 Thoroughfare</p> <p>2 Locality</p> <p>1 Administrative area</p> <p>0 None</p>
Latitude	Eight-digit number in degrees and calculated to five decimal places (in the format specified).
Longitude	Eight-digit number in degrees and calculated to five decimal places (in the format specified).
SearchDistance	The radius of accuracy in meters, providing an indication of the probable maximum distance between the given geocode and the actual physical location. This field is derived from and dependent upon the accuracy and coverage of the underlying reference data.

Table 124: City/Street/Postal Code Centroid Match Codes

Element	Match Code
Address Point	P4
Address Point Interpolated	I4
Street Centroid	A4/P3
Postal Code/City Centroid	A3/P2/A2

Note: Geocode.Match.Code does not return two coordinates for a street segment (such as the beginning and ending of a portion of a street). Instead, with input resulting in return codes of I3 (interpolated to thoroughfare or street level, where no input premise number was provided), the complete street is used in the computation.

Result Indicators

Result indicators provide information about the kinds of processing performed on an address. There are two types of result indicators:

Record-Level Result Indicators

Record-level result indicators provide data about the results of Validate Address Loqate processing for each record, such as the success or failure of the match attempt, which coder processed the address, and other details. The following table lists the record-level result indicators returned by Validate Address Loqate.

Table 125: Record Level Indicators

Field Name	Description
Confidence	The level of confidence assigned to the address being returned. Range is from zero (0) to 100; zero indicates failure, 100 indicates a very high level of confidence that the match results are correct. For multiple matches, the confidence level is 0. For details about how this number is calculated, see Introduction to the Validate Address Confidence Algorithm .

Field Name	Description
CouldNotValidate	<p>If no match was found, which address component could not be validated:</p> <ul style="list-style-type: none"> • ApartmentNumber • HouseNumber • StreetName • PostalCode • City • Directional • StreetSuffix • Firm • POBoxNumber <p>Note: More than one component may be returned, in a comma-separated list.</p>
MatchScore	<p>MatchScore provides an indication of the similarity between the input data and the closest reference data match. It is significantly different from Confidence in that Confidence indicates how much the input address changed to obtain a match, whereas the meaning of Match Score varies between U.S. and non-U.S. addresses.</p> <p>The int getFieldMatchscore (unit record, const char*) field is a decimal value between 0 and 100 that reflects the similarity between the identified input data and the closest reference data match. A result of 100 indicates that no changes other than alias, casing, or diacritic changes have been made to the input data. A result of 0 indicates that there is no similarity between the input data and closest reference data match.</p> <p>Note: The Validate Address Loqate and Advanced Matching Module components both use the MatchScore field. The MatchScore field value in the output of a dataflow is determined by the last stage to modify the value before it is sent to an output stage. If you have a dataflow that contains Validate Address Loqate and Advanced Matching Module components and you want to see the MatchScore field output for each stage, use a Transformer stage to copy the MatchScore value to another field. For example, Validate Address Loqate produces an output field called MatchScore and then a Transformer stage copies the MatchScore field from Validate Address Loqate to a field called AddressMatchScore. When the matcher stage runs it populates the MatchScore field with the value from the matcher and passes through the AddressMatchScore value from Validate Address Loqate.</p>
ProcessedBy	<p>Which address coder processed the address:</p> <p>LOQATE The Loqate coder processed the address.</p>

Field Name	Description				
Status	<p>Reports the success or failure of the match attempt. For multiple matches, this field is "F" for all the possible matches.</p> <table> <tr> <td>null</td> <td>Success</td> </tr> <tr> <td>F</td> <td>Failure</td> </tr> </table>	null	Success	F	Failure
null	Success				
F	Failure				
Status.Code	<p>Reason for failure, if there is one.</p> <ul style="list-style-type: none"> • UnableToValidate 				
Status.Description	<p>Description of the problem, if there is one.</p> <table> <tr> <td>Address Not Found</td> <td>This value will appear if Status.Code=UnableToValidate.</td> </tr> </table>	Address Not Found	This value will appear if Status.Code=UnableToValidate.		
Address Not Found	This value will appear if Status.Code=UnableToValidate.				

Field-Level Result Indicators

Field-level result indicators describe how Validate Address Loqate handled each address element. Field-level result indicators are returned in the qualifier "Result". For example, the field-level result indicator for HouseNumber is contained in **HouseNumber.Result**.

To enable field-level result indicators, check the **Include result codes for individual fields** box.

The following table lists the field-level result indicators. If a particular field does not apply to an address, the result indicator may be blank.

Table 126: Field-Level Result Indicators

Field Name	Description
ApartmentLabel.Result	A Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C Corrected. U.S. and Canadian addresses only.
	F Formatted. The spacing and/or punctuation was changed to conform to postal standards.
	P Pass-through. The data was not used in the validation process, but it was preserved in the output. U.S. and Canadian addresses only.
	R The apartment label is required but is missing from the input address. U.S. addresses only.
	S Standardized. This option includes any standard abbreviations.
	U Unmatched. Does not apply to Canadian addresses.
	V Validated. The data was confirmed correct and remained unchanged from input.
ApartmentNumber.Result	A Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C Corrected. Canadian addresses only.
	F Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	P Pass-through. The data was not used in the validation process, but it was preserved in the output. U.S. addresses that are an EWS match will have a value of P. U.S. and Canadian addresses only.
	R The apartment number is required but is missing from the input address. U.S. addresses only.
	S Standardized. This option includes any standard abbreviations. Does not apply to U.S. addresses.
	U Unmatched.
	V Validated. The data was confirmed correct and remained unchanged from input.

Field Name	Description
City.Result	A Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C Corrected. U.S. and Canadian addresses only.
	F Hyphens missing or punctuation errors. Canadian addresses only.
	M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. Does not apply to U.S. or Canadian addresses.
	P Pass-through. The data was not used in the validation process, but it was preserved in the output.
	R The city is required but is missing from the input address. U.S. addresses only.
	S Standardized. This option includes any standard abbreviations. Does not apply to U.S. addresses.
	V Validated. The data was confirmed correct and remained unchanged from input.
Country.Result	These result codes do not apply to U.S. or Canadian addresses.
	M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field.
	S Standardized. This option includes any standard abbreviations.
	U Unmatched.
	V Validated. The data was confirmed correct and remained unchanged from input.
County.Result*	The smallest geographic data element within a country, for instance, USA County
FirmName.Result	C Corrected. U.S. addresses only.
	P Pass-through. The data was not used in the validation process, but it was preserved in the output. U.S. and Canadian addresses only.
	U Unmatched. U.S. and Canadian addresses only.
	V Validated. The data was confirmed correct and remained unchanged from input. U.S. addresses only.

Field Name	Description
HouseNumber.Result	A Appended. The field was added to a blank input field. Canadian addresses only.
	C Corrected. Canadian addresses only.
	F Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	O Out of range. Does not apply to U.S. or Canadian addresses.
	P Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only.
	R The house number is required but is missing from the input address. Canadian addresses only.
	S Standardized. This option includes any standard abbreviations. Does not apply to U.S. or Canadian addresses.
	V Validated. The data was confirmed correct and remained unchanged from input.
LeadingDirectional.Result	A Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C Corrected. Non-blank input was corrected to a non-blank value. U.S. addresses only.
	F Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only.
	P Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only.
	S Standardized. This option includes any standard abbreviations.
	U Unmatched.
	V Validated. The data was confirmed correct and remained unchanged from input. Does not apply to Canadian addresses.

Field Name	Description
POBox.Result	A Appended. The field was added to a blank input field. Canadian addresses only.
	C Corrected. Canadian addresses only.
	F Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	M Multiple matches. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only.
	P Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only.
	R The P.O. Box number is required but is missing from the input address. U.S. addresses only.
	S Standardized. This option includes any standard abbreviations.
	U Unmatched.
V Validated. The data was confirmed correct and remained unchanged from input.	
PostalCode.Result	A Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C Corrected. U.S. and Canadian addresses only.
	F Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. Does not apply to Canadian addresses.
	P Pass-through. The data was not used in the validation process, but it was preserved in the output. Does not apply to U.S. addresses.
	R The postal code is required but is missing from the input address. U.S. addresses only.
	S Standardized. This option includes any standard abbreviations. Does not apply to U.S. or Canadian addresses.
	U Unmatched. For example, if the street name does not match the postal code, both StreetName.Result and PostalCode.Result will contain U.
V Validated. The data was confirmed correct and remained unchanged from input.	

Field Name	Description
PostalCode.Type	<p>P The ZIP Code™ contains only PO Box addresses. U.S. addresses only.</p> <p>U The ZIP Code™ is a unique ZIP Code™ assigned to a specific company or location. U.S. addresses only.</p> <p>M The ZIP Code™ is for military addresses. U.S. addresses only.</p> <p>null The ZIP Code™ is a standard ZIP Code™.</p>
Principality.Result *	The largest geographic data element within a country
StateProvince.Result	<p>A Appended. The field was added to a blank input field. U.S. and Canadian addresses only.</p> <p>C Corrected. U.S. addresses only.</p> <p>M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. Does not apply to U.S. or Canadian addresses.</p> <p>P Pass-through. The data was not used in the validation process, but it was preserved in the output. U.S. and Canadian addresses only.</p> <p>R The state is required but is missing from the input address. U.S. addresses only.</p> <p>S Standardized. This option includes any standard abbreviations. Does not apply to U.S. addresses.</p> <p>U Unmatched. Does not apply to Canadian addresses.</p> <p>V Validated. The data was confirmed correct and remained unchanged from input.</p>
StreetAlias.Result	An alternate name for a street; typically applies only to a specific range of addresses on the street. If you do not allow street aliases in the output then the street's "base" name will appear in the output regardless of whether or not there is an alias for the street. The base name is the name that applies to the entire street. For example: If StreetName is "N MAIN ST" the StreetAlias field would contain "MAIN" and the thoroughfare type,"ST", would be returned in the StreetSuffix field.

Field Name	Description
StreetName.Result	A Appended. The field was added to a blank input field. Canadian addresses only.
	C Corrected. U.S. and Canadian addresses only.
	F Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only.
	P Pass-through. The data was not used in the validation process, but it was preserved in the output. Does not apply to U.S. addresses.
	S Standardized. This option includes any standard abbreviations. U.S. and Canadian addresses only.
	U Unmatched.
	V Validated. The data was confirmed correct and remained unchanged from input.
StreetSuffix.Result	A Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C Corrected. U.S. and Canadian addresses only.
	F Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only.
	P Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only.
	S Standardized. This option includes any standard abbreviations.
	U Unmatched. Does not apply to U.S. addresses.
	V Validated. The data was confirmed correct and remained unchanged from input.
Subcity.Result*	A smaller population center data element, dependent on the contents of the Locality field. For instance, Turkish Neighbourhood .
Substreet.Result*	The dependent street or block data element within a country. For instance, UK Dependent Street .

Field Name	Description
TrailingDirectional.Result	A Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C Corrected. U.S. and Canadian addresses only.
	F Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only.
	P Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only.
	S Standardized. This option includes any standard abbreviations.
	U Unmatched. Does not apply to Canadian addresses.
	V Validated. The data was confirmed correct and remained unchanged from input.

*This is a subfield and may not contain data.

The AVC Code

The *Address Verification Code (AVC)* is an 11-byte code that is made up of accuracy indicators for addresses; the codes tell you the quality of the processing results and provide guidelines on how to correct the input data if necessary. Each individual address receives its own code. This code is automatically returned within your dataflow output. An example of an AVC is:

V44-I44-P6-100

An AVC has eight parts:

- Verification Status
- Post-Process Verification Match Level
- Pre-Process Verification Match Level
- Parsing Status
- Lexicon Identification Match Level
- Context Identification Match Level
- Postcode Status
- Matchscore

Verification Status

The level to which an address was verified.

- **V**—Verified. A complete match was made between the input data and a single record from the available reference data. For simple address validation, this is considered the best code to return.
- **P**—Partially verified. A partial match was made between the input data and a single record from the available reference data. This could mean that there is granular data for the address information that was provided, but additional information is required to return a full validation.
- **A**—Ambiguous. There are multiple addresses that could match the input.
- **U**—Unable to verify. This gets returned when there is not enough information to verify an address or when the input query is unreadable. The output fields will contain the input data.
- **R**—Reverted. The record could not be verified to the specified minimum acceptable level. This occurs when advanced options such as minimum reversion levels are set on a process. The output fields will contain the input data.
- **C**—Conflict. There is more than one close reference data match with conflicting values.

Post-Process Verification Match Level

The level to which the input data matches the available reference data after processing.

- **5**—Delivery point (building or post box). The record will be passed or will have high confidence if ApartmentNumber, HouseNumber, Street, City, and StateProvince supplied in the input record match to the Loqate reference dataset. Will have moderate confidence if ApartmentNumber is correct but other remaining fields are incorrect, but in this case the Loqate engine should be able to identify the ApartmentNumber as ApartmentNumber is at a more granular level. It will have zero confidence if ApartmentNumber and other fields are unable to be parsed by the Loqate engine.
- **4**—Premise or building. The record will be passed or will have high confidence if House Number, Street, City, and StateProvince supplied in the input record match the Loqate reference dataset. Will have moderate confidence if HouseNumber is correct but the other fields are not; however, in this case the Loqate engine should be able to identify the HouseNumber because HouseNumber is at a more granular level. It will have zero confidence if the HouseNumber and other fields are unable to be parsed by the Loqate engine.
- **3**—Thoroughfare, road, or street. The record will be passed or will have high confidence if Street, City, and StateProvince supplied in the input record match the Loqate reference dataset. Will have moderate confidence if City is correct but StateProvince is not; however, in this case the Loqate engine should be able to identify the StateProvince as City itself is the part of StateProvince. It will have zero confidence if City or both fields (City and State Province) are unable to be parsed by the Loqate engine.
- **2**—Locality (city or town). The record will be passed or will have high confidence if both City and StateProvince supplied in the input record match the Loqate reference dataset. Will have moderate confidence if City is correct but StateProvince is not; however, in this case the Loqate Engine should be able to identify the StateProvince as City itself is the part of StateProvince. It will have zero confidence if City or both fields (City and StateProvince) are unable to be parsed by the Loqate engine.
- **1**—Administrative area (state or region). The record will be passed or will have high confidence if the StateProvince supplied in the input record matches the Loqate reference dataset.
- **0**—None. This is equivalent to loosest match option.

Pre-Process Verification Match Level

The level to which the input data matches the available reference data before processing.

- **5**—Delivery point (building or post box)
- **4**—Premise or building.
- **3**—Thoroughfare, road, or street.
- **2**—Locality (city or town).
- **1**—Administrative area (state or region).
- **0**—None.

Parsing Status

The level to which an address was parsed.

- **I**—Identified and parsed. The input data has been identified and placed into components. For example, with "123 Kingston Av" Validate Address Loqate would be able to determine that "123" was a Premise Number, "Kingston" was the Thoroughfare Name, and "Av" or "Avenue" would be the Thoroughfare Type.
- **U**—Unable to parse. Validate Address Loqate was unable to identify and parse the input data. As with the "Unverified" verification status, the input data was incomplete or vague.

Lexicon Identification Match Level

The level to which the input data has some recognized form through the use of pattern matching (for instance, a numeric value could be a premise number) and lexicon matching (for example, "rd" could be Thoroughfare Type "road"; "London" could be a locality, and so on).

- **5**—Delivery point (building or post box)
- **4**—Premise or building.
- **3**—Thoroughfare, road, or street.
- **2**—Locality (city or town).
- **1**—Administrative area (state or region).
- **0**—None.

Context Identification Match Level

The level to which the input data can be recognized based on the context in which it appears. This is the least accurate form of matching and is based on identifying a word as a particular address element. For example, input could be determined to be a thoroughfare because it was preceded by something that could be a premise and followed by something that could be a locality, the latter items being identified through a match against the reference data or the lexicon.

- **5**—Delivery point (building or post box)
- **4**—Premise or building.

- **3**—Thoroughfare, road, or street.
- **2**—Locality (city or town).
- **1**—Administrative area (state or region).
- **0**—None.

Postcode Status

The level to which a postal code was verified.

- **P8**—PostalCodePrimary and PostalCodeSecondary verified.
- **P7**—PostalCodePrimary verified, PostalCodeSecondary added or changed.
- **P6**—PostalCodePrimary verified.
- **P5**—PostalCodePrimary verified with small change.
- **P4**—PostalCodePrimary verified with large change.
- **P3**—PostalCodePrimary added.
- **P2**—PostalCodePrimary identified by lexicon.
- **P1**—PostalCodePrimary identified by context.
- **P0**—PostalCodePrimary empty.

Match Score

A numeric value between 0 and 100 representing the similarity between the identified input data and the output data for the record. A result of 100 means that no changes other than additions, alias, casing, or diacritic changes have been made to the input data. A result of 0 means there is no similarity between the input data item and the output data provided.

AMAS Output

The following table lists the standard fields that are output by Validate Address Loqate.

Table 127: Output Fields

Field Name	Description
Barcode	Standard barcode based on the DPID.
	F Failure (no barcode found)
	20-digit number Success

Field Name	Description
DPID	The Delivery Point Identifier. An eight-digit number from the Australia Post Postal Address File that uniquely identifies a mail delivery point, such as a street address. Note: This field will contain "00000000" for Australian addresses that are not AMAS-verified and will be empty for non-Australian addresses.
FloorNumber	The floor/level number, for example: 123 E Main St Apt 3, 4th Floor
FloorType	The floor/level type, for example: 123 E Main St Apt 3, 4th Floor
PostalBoxNum	The postal delivery number, for example: PO Box 42

Universal Name Stages

Name Parser (DEPRECATED)

Attention: The Name Parser stage is deprecated and may not be supported in future releases. Use Open Name Parser for parsing names.

Name Parser breaks down personal and business names and other terms in the name data field into their component parts. The parsing process includes an explanation of the function, form and syntactical relationship of each part to the whole. These parsed name elements are then subsequently available to other automated operations such as name matching, name standardization, or multirecord name consolidation.

Name parsing does the following:

- Determines the entity type of a name in order to describe the function which the name performs. Name entity types are divided into two major groupings: Personal names and business names with subgroups within these major groupings.
- Determines the form of a name in order to understand which syntax the parser should follow for parsing. Personal names usually take on a natural (signature) order or a reverse order. Business names are usually ordered hierarchically.
- Determines and labels the component parts of a name so that the syntactical relationship of each name part to the entire name is identified. The personal name syntax includes prefixes, first, middle, and last name parts, suffixes and account description terms among other personal name parts.

The business name syntax includes the primary text, insignificant terms, prepositions, objects of the preposition and suffix terms among other business name parts.

- Determines the gender of the name. The gender is determined based on cultural assumptions which you specify. For example, Jean is a male name in France but a female name in the U.S. If you know the names you are processing are from France, you could specify French as the gender determination culture. The Name Parser uses data from the First Name and Compound First Names tables to determine gender. If a name is not found in either table and a title is present in the name, the parser checks the Title table to determine gender. Otherwise, the gender is marked as unknown.

Note: If a field on your input record already contains one of the supported cultures, you can predefine the GenderDeterminationSource field in your input to override the Gender Determination Source in the GUI.

- Assigns a parsing score which indicates the degree of confidence which the parser has that its parsing is correct.

Input

Attention: The Name Parser stage is deprecated and may not be supported in future releases. Use Open Name Parser for parsing names.

Table 128: Name Parser Input

Field Name	Description / Valid Values																										
GenderDeterminationSource	<p>The culture of the name data to use to determine gender. Default uses cross-cultural rules. For example, Jean is commonly a female name and Default identifies it as such, but it is identified as a male name if you select French. The options are listed below along with example countries for each culture. Note that the list of countries under each culture is not exhaustive.</p> <table border="0"> <tr> <td>SLAVIC</td> <td>Bosnia, Poland, Albania.</td> </tr> <tr> <td>ARMENIAN</td> <td>Armenia.</td> </tr> <tr> <td>DEFAULT</td> <td>Bulgaria, Cayman Islands, Ireland, U.S., U.K.</td> </tr> <tr> <td>FRENCH</td> <td>France.</td> </tr> <tr> <td>SCANDINAVIAN</td> <td>Denmark, Finland, Iceland, Norway, Sweden.</td> </tr> <tr> <td>GERMANIC</td> <td>Austria, Germany, Luxembourg, Switzerland, The Netherlands.</td> </tr> <tr> <td>GREEK</td> <td>Greece.</td> </tr> <tr> <td>HUNGARIAN</td> <td>Hungary.</td> </tr> <tr> <td>ITALIAN</td> <td>Italy.</td> </tr> <tr> <td>PORTUGUESE</td> <td>Portugal.</td> </tr> <tr> <td>ROMANIA</td> <td>Romania.</td> </tr> <tr> <td>HISPANIC</td> <td>Spain.</td> </tr> <tr> <td>ARABIC</td> <td>Tunisia.</td> </tr> </table> <p>GenderDeterminationSource is also used by Name Variant Finder to limit the returned name variations based on culture. For more information, see Name Variant Finder on page 770.</p>	SLAVIC	Bosnia, Poland, Albania.	ARMENIAN	Armenia.	DEFAULT	Bulgaria, Cayman Islands, Ireland, U.S., U.K.	FRENCH	France.	SCANDINAVIAN	Denmark, Finland, Iceland, Norway, Sweden.	GERMANIC	Austria, Germany, Luxembourg, Switzerland, The Netherlands.	GREEK	Greece.	HUNGARIAN	Hungary.	ITALIAN	Italy.	PORTUGUESE	Portugal.	ROMANIA	Romania.	HISPANIC	Spain.	ARABIC	Tunisia.
SLAVIC	Bosnia, Poland, Albania.																										
ARMENIAN	Armenia.																										
DEFAULT	Bulgaria, Cayman Islands, Ireland, U.S., U.K.																										
FRENCH	France.																										
SCANDINAVIAN	Denmark, Finland, Iceland, Norway, Sweden.																										
GERMANIC	Austria, Germany, Luxembourg, Switzerland, The Netherlands.																										
GREEK	Greece.																										
HUNGARIAN	Hungary.																										
ITALIAN	Italy.																										
PORTUGUESE	Portugal.																										
ROMANIA	Romania.																										
HISPANIC	Spain.																										
ARABIC	Tunisia.																										
Name	The name you want to parse. This field is required.																										

Options

Attention: The Name Parser stage is deprecated and may not be supported in future releases. Use Open Name Parser for parsing names.

To specify the Name Parser options, double-click the instance of Name Parser on the canvas. The Name Parser Options dialog displays.

Table 129: Name Parser Options

Option	Description
Parse personal names	Check this box to parse personal names.
Separate conjoined names into multiple records Select a match results in the Match Results List and then click Remove .	<p>Click this box to separate names containing more than one individual into multiple records, for example, <i>Bill & Sally Smith</i>.</p> <p>When a conjoined record results in two separate name records, a Parser Record ID output field is generated. Each pair of separate name records are identified with the same Parser Record ID.</p>
Gender Determination Source Select a match results in the Match Results List and then click Remove .	<p>Determines how the Name Parser assigns a gender to the name. For most cases, Default is the best setting because it covers a wide variety of names. If you are processing names from a specific culture, select that culture. Selecting a specific culture helps ensure that the proper gender is assigned to the names. For example, if you leave Default selected, then the name Jean is identified as a female name. If you select French, it is identified as a male name.</p> <p>Note: If you select a culture but the name is not found in that culture, gender is determined using the Default culture, which includes data from a variety of cultures.</p>
Order	<p>Specifies how the name fields are ordered in your input records. One of the following:</p> <p>Natural The name fields are ordered by Title, First Name, Middle Name, Last Name, and Suffix.</p> <p>Reverse The name fields are ordered by Last Name first.</p> <p>Mixed The name fields are ordered using a combination of natural and reverse.</p>
Retain Periods	Retains punctuation in the parsed personal name field.
Parse Business Names	Check this box to parse business names.
Retain Periods	Check this box to return punctuation to the parsed business name field.

Option	Description
User-Defined Table	Click any of the User-Defined Tables to add values to existing values in the various parser tables. This capability enables you to customize tables for your unique business environment. Click Configure to select an XML file that contains the values that you want to add. For more information about user-defined tables, see Modifying Name Parser User-Defined Tables on page 753.

Modifying Name Parser User-Defined Tables

Attention: The Name Parser stage is deprecated and may not be supported in future releases. Use Open Name Parser for parsing names.

You can add, modify, and delete values in the Name Parser tables to customize them for your unique business environment.

Name Parser's user-defined tables are XML files located by default in the <Drive>:\Program Files\Precisely\Spectrum\server\modules\parser\data folder. Spectrum Technology Platform includes the following user-defined tables:

UserAccountDescriptions.xml

Table 130: UserAccountDescriptions.xml Columns

Column Name	Description / Valid Values
LookupValue	A lookup term commonly found in an Account Description. Any single-word text. Case insensitive.

Example entry:

```
<table-data>
  <deleted-entries delimiter-character="|">
    <deleted-entry-group>
      <![CDATA[
        LookupValue
        ART
        AND
      ]]>
    </deleted-entry-group>
  </deleted-entries>
```

```

<added-entries delimiter-character="|">
  <![CDATA[
    LookupValue
    A/C
    ACCOUNT
    EXP
  ]]>
</added-entries>
</table-data>

```

UserCompanyPrepositions.xml

Table 131: UserCompanyPrepositions.xml Columns

Column Name	Description / Valid Values
LookupValue	Any preposition (for example, "of" or "on") commonly found in company names. Any single-word text. Case insensitive.

Example entry:

```

<table-data>
  <deleted-entries delimiter-character="|">
    <deleted-entry-group>
      <![CDATA[
        LookupValue
        AROUND
        NEAR
      ]]>
    </deleted-entry-group>
  </deleted-entries>
  <added-entries delimiter-character="|">
    <![CDATA[
      LookupValue
      ABOUT
      AFTER
      ACROSS
    ]]>
  </added-entries>
</table-data>

```

*UserCompanySuffixes.xml***Table 132: UserCompanySuffixes.xml Columns**

Column Name	Description / Valid Values
LookupValue	Any suffix commonly found in company names. Examples include "Inc." and "Co." Any single-word text. Case insensitive.

Example entry:

```
<table-data>
  <deleted-entries delimiter-character="|">
    <deleted-entry-group>
      <![CDATA[
        LookupValue
        SANDY
        CLUE
      ]]>
    </deleted-entry-group>
  </deleted-entries>
  <added-entries delimiter-character="|">
    <![CDATA[
      LookupValue
      LTD
      LLC
      CO
      INC
    ]]>
  </added-entries>
</table-data>
```

*UserCompanyTerms.xml***Table 133: UserCompanyTerms.xml Columns**

Column Name	Description / Valid Values
LookupValue	Any term commonly found in a company name. Any single-word text. Case insensitive.

Example entry:

```
<table-data>
  <deleted-entries delimiter-character="|">
    <deleted-entry-group>
      <![CDATA[
        LookupValue
        MARY
        BLUE
      ]]>
    </deleted-entry-group>
  </deleted-entries>
  <added-entries delimiter-character="|">
    <![CDATA[
      LookupValue
      ARC
      ARCADE
      ASSEMBLY
      ARIZONA
    ]]>
  </added-entries>
</table-data>
```

UserCompoundFirstNames.xml

This table contains user-defined compound first names. Compound names are names that consist of two words.

Table 134: UserCompoundFirstNames.xml Columns

Column Name	Description / Valid Values
FirstName	The compound first name. Maximum of two words. Case insensitive.
Culture	The culture in which this FirstName/Gender combination applies. You may use any of the values that are valid in the GenderDeterminationSource input field. For more information, see Input on page 750.

Column Name	Description / Valid Values
Gender	<p>The gender most commonly associated with this FirstName/Culture combination. One of the following:</p> <p>M The name is a male name.</p> <p>F The name is a female name.</p> <p>A Ambiguous. The name can be either male or female.</p> <p>U Unknown. The gender of this name is not known. Unknown is assumed if this field is left blank.</p>
Frequency	Not used in this release. You may leave this column blank.

Example entry:

```

<table-data>
  <deleted-entries delimiter-character="|">
    <deleted-entry-group>
      <![CDATA[
        FirstName
        ANN MARIE
        BILLY JOE
      ]]>
    </deleted-entry-group>
    <deleted-entry-group>
      <![CDATA[
        FirstName|Frequency
        KAREN SUE|0.126
        BILLY JOE|0.421
      ]]>
    </deleted-entry-group>
    <deleted-entry-group>
      <![CDATA[
        FirstName|Gender|Culture
        JEAN ANN|M|DEFAULT
        JEAN CLUADE|F|FRENCH
      ]]>
    </deleted-entry-group>
  </deleted-entries>
  <added-entries delimiter-character="|">
    <![CDATA[
      FirstName|Gender|Culture
      JOHN Henry|M|DEFAULT
      A'SHA A'MAR|F|ARABIC
      BILLY JO|A|DEFAULT
    ]]>
  </added-entries>

```

```

    ]]>
  </added-entries>
</table-data>

```

UserConjunctions.xml

This table contains a list of user-defined conjunctions, such as "and", "or", or "&".

Table 135: UserConjunctions.xml Columns

Column Name	Description / Valid Values
LookupValue	Any conjunction. Must be a single word. Case insensitive.

Example entries:

```

<table-data>
  <deleted-entries delimiter-character="|">
    <deleted-entry-group>
      <![CDATA[
        LookupValue
        FIND
        CARE
        %
      ]]>
    </deleted-entry-group>
  </deleted-entries>
  <added-entries delimiter-character="|">
    <![CDATA[
      LookupValue
      &
      AND
      OR
    ]]>
  </added-entries>
</table-data>

```

*UserFirstNames.xml***Table 136: UserFirstNames.xml Columns**

Column Name	Description / Valid Values
FirstName	The first name described by this table row. Case insensitive.
Gender	The gender most commonly associated with this FirstName/Culture combination. One of the following: M The name is a male name. F The name is a female name. A Ambiguous. The name can be either male or female. U Unknown. The gender of this name is not known. Unknown is assumed if this field is left blank.
Culture	The culture in which this FirstName/Gender combination applies. You may use any of the values that are valid in the GenderDeterminationSource input field. For more information, see Input on page 750.

Example entry:

```
<table-data>
  <deleted-entries delimiter-character="|">
    <deleted-entry-group>
      <![CDATA[
        FirstName
        AADEL
        AADIL
      ]]>
    </deleted-entry-group>
    <deleted-entry-group>
      <![CDATA[
        FirstName
        A ' SACE
        A ' BOCKETT
      ]]>
    </deleted-entry-group>
    <deleted-entry-group>
      <![CDATA[
        FirstName|Gender|Culture
```

```

                ALII|M|DEFAULT
                AISHA|F|ARABIC
            ]]>
        </deleted-entry-group>
    <deleted-entry-group>
        <![CDATA[
            FirstName|Gender
            JOHE|M
        ]]>
    </deleted-entry-group>
</deleted-entries>
<added-entries delimiter-character="|">
    <![CDATA[
        FirstName|Gender|Culture
        JOHE|M|DEFAULT
        A'SHAN|F|ARABIC
    ]]>
</added-entries>
</table-data>

```

UserGeneralSuffixes.xml

This table contains a list of user-defined suffixes used in personal names that are not maturity suffixes, such as "MD" or "PhD".

Table 137: UserGeneralSuffixes.xml Columns

Column Name	Description / Valid Values
LookupValue	Any suffix that is frequently applied to personal names and is not a maturity suffix. Must be a single word. Case insensitive.

Example entry:

```

<table-data>
  <deleted-entries delimiter-character="|">
    <deleted-entry-group>
      <![CDATA[
        LookupValue
        AND
        WILL
        TUNA
      ]]>
    </deleted-entry-group>

```



```

</deleted-entries>
<added-entries delimiter-character="|" ">
  <![CDATA[
    LookupValue
    ACCOUNTANT
    ATTORNEY
    ANALYST
    ASSISTANT
  ]]>
</added-entries>
</table-data>

```

UserLastNamePrefixes.xml

This table contains a list of user-defined prefixes that occur in a person's last name such as "Van", "De", or "La".

Table 138: UserLastNamePrefixes.xml Columns

Column Name	Description / Valid Values
LookupValue	Any prefix that occurs as part of an individual's last name. Any single-word text. Case insensitive.

Example entry:

```

<table-data>
  <deleted-entries delimiter-character="|" ">
    <deleted-entry-group>
      <![CDATA[
        LookupValue
        DO
        RUN
        ANIMAL
      ]]>
    </deleted-entry-group>
  </deleted-entries>
  <added-entries delimiter-character="|" ">
    <![CDATA[
      LookupValue
      D'
      DA
      DEN
      DEL
    ]]>
  </added-entries>
</table-data>

```

```

]]>
</added-entries>
</table-data>

```

UserLastNames.xml

Table 139: UserLastNames.xml Columns

Column Name	Description / Valid Values
LastName	The last name described by this table row. Case insensitive.
Gender	<p>The gender most commonly associated with this FirstName/Culture combination. One of the following:</p> <p>M The name is a male name.</p> <p>F The name is a female name.</p> <p>A Ambiguous. The name can be either male or female.</p> <p>U Unknown. The gender of this name is not known. Unknown is assumed if this field is left blank.</p>
Culture	The culture in which this FirstName/Gender combination applies. You may use any of the values that are valid in the GenderDeterminationSource input field. For more information, see Input on page 750.

Example entry:

```

<table-data>
  <deleted-entries delimiter-character="|">
    <deleted-entry-group>
      <![CDATA[
        LastName
        Rusod
        AADIL
      ]]>
    </deleted-entry-group>
    <deleted-entry-group>
      <![CDATA[
        LastName

```

```

        KAASEEY
        JOIEN
    ]]>
</deleted-entry-group>
</deleted-entries>
<added-entries delimiter-character="|">
    <![CDATA[
        LastName|Culture|Gender
        SMITH|ENGLISH|A
        WILSON|ENGLISH|A
        JONES|ENGLISH|A
    ]]>
</added-entries>
</table-data>

```

UserMaturitySuffixes.xml

This table contains user-defined generational suffixes used in a person's name, such as "Jr." or "Sr."

Table 140: UserMaturitySuffixes.xml Columns

Column Name	Description / Valid Values
LookupValue	A generational suffix used in personal names. Any single-word text. Case insensitive.

Example entry:

```

<table-data>
  <deleted-entries delimiter-character="|">
    <deleted-entry-group>
      <![CDATA[
        LookupValue
        I
        V
        18
        VI
      ]]>
    </deleted-entry-group>
  </deleted-entries>
  <added-entries delimiter-character="|">
    <![CDATA[
        LookupValue
        I
        II
    ]]>
  </added-entries>
</table-data>

```

```

        III
      ]]>
    </added-entries>
  </table-data>

```

UserTitles.xml

This table contains user-defined titles used in a person's name, such as "Mr." or "Ms."

Table 141: UserTitles.xml Columns

Column Name	Description / Valid Values
LookupValue	A title used in personal names. Any single-word text. Case insensitive.
Gender	The gender most commonly associated with this title. One of the following: <ul style="list-style-type: none"> M The name is a male name. F The name is a female name. A Ambiguous. The name can be either male or female. U Unknown. The gender of this name is not known. Unknown is assumed if this field is left blank.

Example entry:

```

<table-data>
  <deleted-entries delimiter-character="|">
    <deleted-entry-group>
      <![CDATA[
        LookupValue
        Belt
        Friend
        Thursday
        Red
      ]]>
    </deleted-entry-group>
  </deleted-entries>
  <added-entries delimiter-character="|">
    <![CDATA[
      LookupValue|Gender
      Mrs|F
    ]]>
  </added-entries>
</table-data>

```

```

        Mr|M
        Most|F
    ]]>
</added-entries>
</table-data>

```

Sample User-Defined Table

The figure below shows a sample UserFirstNames.xml table and the syntax to use when modifying user-defined tables.

```

<table-data>
  <deleted-entries delimiter-character="|">
    <deleted-entry-group>
      <![CDATA[
        FirstName
        AADEL
        AADIL
      ]]>
    </deleted-entry-group>
    <deleted-entry-group>
      <![CDATA[
        FirstName|Frequency
        A'SACE|0.126
        A'BECKETT|0.421
      ]]>
    </deleted-entry-group>
    <deleted-entry-group>
      <![CDATA[
        FirstName|Gender|Culture|VariantGroup
        ALI|M|DEFAULT|GROUP88
        AISHA|F|ARABIC|GROUP43
      ]]>
    </deleted-entry-group>
    <deleted-entry-group>
      <![CDATA[
        FirstName|Gender
        JOHN|M
      ]]>
    </deleted-entry-group>
  </deleted-entries>
  <added-entries delimiter-character="|">
    <![CDATA[
      FirstName|Gender|Culture
      JOHN|M|DEFAULT
      A'SHA|F|ARABIC
      JAMES|M|DEFAULT
    ]]>
  </added-entries>

```

```
</table-data>
```

Output

Attention: The Name Parser stage is deprecated and may not be supported in future releases. Use Open Name Parser for parsing names.

Table 142: Name Parser Output

Field Name	Format	Description / Valid Values
AccountDescription	String	An account description that is part of the name. For example, in "Mary Jones Account # 12345", the account description is "Account#12345".
EntityType	String	Indicates the type of name. One of the following: Firm The name is a company name. Personal The name is an individual person's name.
Fields Related to Names of Companies		
FirmModifier.1.Object	String	The first object of a preposition occurring in firm name. For example, in the firm name "Pratt & Whitney Division of United Technologies", the first object of a preposition is "United Technologies".
FirmModifier.1.Preposition	String	The first preposition occurring in firm name. For example, in the firm name "Pratt & Whitney Division of United Technologies", "of" would be the first preposition.
FirmModifier.2.Object	String	The second object of a preposition occurring in firm name. For example, in the firm name "Church of Our Lady of Lourdes", the second object of a preposition is the second "Lourdes".
FirmModifier.2.Preposition	String	The second preposition occurring in firm name. For example, in the firm name "Church of Our Lady of Lourdes", the second preposition is the second "of".

Field Name	Format	Description / Valid Values
FirmName	String	The name of a company. For example, "Precisely Software, Inc."
FirmPrimary	String	The base part of a company's name. For example, "Precisely Software".
FirmSuffix	String	The corporate suffix. For example, "Co." and "Inc."
Fields Related to Names of Individual People		
FirstName	String	The first name of a person.
FirstNameVariantGroup	String	<p>A numeric ID that indicates the group of similar names to which first name belongs. For example, Muhammad, Mohammed, and Mehmet all belong to the same Name Variant Group. The actual group ID is assigned when the add-on data is loaded.</p> <p>This field is only populated if you have purchased the Name Variant Group feature.</p>
GenderCode	String	<p>A person's gender as determined by analyzing the first name. One of the following:</p> <p>A Ambiguous. The name is both a male and a female name. For example, Pat.</p> <p>F Female. The name is a female name.</p> <p>M Male. The name is a male name.</p> <p>U Unknown. The name could not be found in the gender table.</p>
GenderDeterminationSource	String	The culture used to determine a name's gender. If the name could not be found in the gender table, this field is blank.
GeneralSuffix	String	A person's general/professional suffix. For example, MD or PhD.
LastName	String	The last name of a person.

Field Name	Format	Description / Valid Values
MaturitySuffix	String	A person's maturity/generational suffix. For example, Jr. or Sr.
MiddleName	String	The middle name of a person.
NameScore	String	Score representing quality of the parsing operation, from 0 to 100. 0 indicates poor quality and 100 indicates high quality.
ParserRecordID	String	A unique ID assigned to each input record.
TitleOfRespect	String	A person's title, such as Mr., Mrs., Dr., or Rev.
Fields Related to Conjoined Names		
PersonalName.2.FirstName	String	The first name of the second person in a conjoined name. An example of a conjoined name is "John and Jane Smith."
PersonalName.2.FirstNameVariantGroup	String	<p>A numeric ID that indicates the group of similar names to which first name of the second person in a conjoined name belongs. For example, Muhammad, Mohammed, and Mehmet all belong to the same Name Variant Group. The actual group ID is assigned when the add-on data is loaded.</p> <p>This field is only populated if you have purchased the Name Variant Group feature.</p>
PersonalName.2.GenderCode	String	<p>The gender of the second person in a conjoined name as determined by Name Parser analyzing the first name. An example of a conjoined name is "John and Jane Smith." One of the following:</p> <p>A Ambiguous. The name is both a male and a female name. For example, Pat.</p> <p>F Female. The name is a female name.</p> <p>M Male. The name is a male name.</p> <p>U Unknown. The name could not be found in the gender table.</p>

Field Name	Format	Description / Valid Values
PersonalName.2.GenderDeterminationSource	String	The culture used to determine the gender of the second person in a conjoined name. An example of a conjoined name is "John and Jane Smith."
PersonalName.2.GeneralSuffix	String	The general/professional suffix of the second person in a conjoined name. An example of a conjoined name is "John and Jane Smith." Examples of general suffixes are MD and PhD.
PersonalName.2.LastName	String	The last name of the second person in a conjoined name. An example of a conjoined name is "John and Jane Smith."
PersonalName.2.MaturitySuffix	String	The maturity/generational suffix of the second person in a conjoined name. An example of a conjoined name is "John and Jane Smith." Examples of maturity suffixes are Jr. and Sr.
PersonalName.2.MiddleName	String	The middle name of the second person in a conjoined name. An example of a conjoined name is "John and Jane Smith."
PersonalName.2.TitleOfRespect	String	The title of respect for the second name in a conjoined name. For example, "Mr. and Mrs. Smith" is a conjoined name. Examples of titles of respect are Mr., Mrs., and Dr.
PersonalName.3.FirstName	String	The first name of the third person in a conjoined name. For example, "Mr. & Mrs. John Smith & Dr. Mary Jones" is a conjoined name.
PersonalName.3.FirstNameVariantGroup	String	<p>A numeric ID that indicates the group of similar names to which first name of the second person in a conjoined name belongs. For example, Muhammad, Mohammed, and Mehmet all belong to the same Name Variant Group. The actual group ID is assigned when the add-on data is loaded.</p> <p>This field is only populated if you have purchased the Name Variant Group feature.</p>

Field Name	Format	Description / Valid Values
PersonalName.3.GenderCode	String	<p>The gender of the third person in a conjoined name as determined by Name Parser analyzing the first name. An example of a conjoined name is "Mr. & Mrs. John Smith & Adam Jones". One of the following:</p> <p>A Ambiguous. The name is both a male and a female name. For example, Pat.</p> <p>F Female. The name is a female name.</p> <p>M Male. The name is a male name.</p> <p>U Unknown. The name could not be found in the gender table.</p>
PersonalName.3.GenderDeterminationSource	String	The culture used to determine the gender of the third person in a conjoined name. "Mr. & Mrs. John Smith & Adam Jones".
PersonalName.3.GeneralSuffix	String	The general/professional suffix of the third person in a conjoined name. An example of a conjoined name is "Mr. & Mrs. John Smith & Adam Jones PhD." Examples of general suffixes are MD and PhD.
PersonalName.3.LastName	String	The last name for the third person in a conjoined name. For example, "Mr. & Mrs. John Smith & Dr. Mary Jones" is a conjoined name.
PersonalName.3.MaturitySuffix	String	The maturity/generational suffix of the third person in a conjoined name. An example of a conjoined name is "Mr. & Mrs. John Smith & Adam Jones Sr." Examples of maturity suffixes are Jr. and Sr.
PersonalName.3.MiddleName	String	The middle name for the third person in a conjoined name. For example, "Mr. & Mrs. John Smith & Dr. Mary Jones" is a conjoined name.
PersonalName.3.TitleOfRespect	String	The title of respect for the third name in a conjoined name. For example, "Mr. & Mrs. John Smith & Dr. Mary Jones" is a conjoined name. Examples of titles of respect are Mr., Mrs., and Dr.

Name Variant Finder

Name Variant Finder works in either first name or last name mode to query a database to return alternative versions of a name. For example, "John" and "Jon" are variants for the name "Johnathan". Name Variant Finder requires add-on dictionaries that can be installed using the Universal Name,

Data Normalization, and Advanced Matching database load utility. Contact your sales representative for information on how to obtain these optional culture-specific dictionaries.

Input

Table 143: Name Variant Finder Input Fields

Field Name	Description / Valid Values
FirstName	The name for which you want to find variants, if the name is a given name.
LastName	The name for which you want to find variants, if the name is a surname.
GenderCode	<p>The gender of the name in the FirstName field. One of the following:</p> <p>Note: Gender codes only apply to first names, not last names.</p> <p>M The name is a male name.</p> <p>F The name is a female name.</p> <p>A Ambiguous. The name can be either male or female.</p> <p>U Unknown. The gender of this name is not known.</p>
Ethnicity	<p>The culture most commonly associated with the name in the FirstName or LastName field. You can use the Name Parser or Open Parser stages to populate this field if you do not know the ethnicity for a name.</p> <p>Note: This field was formerly named GenderDeterminationSource.</p>

Options

Table 144: Name Variant Finder Options

Option	Description
First Name	Finds name variations based on first name.

Option	Description
Last Name	Finds name variations based on last name.
Gender Code	Returns the name variations only for the gender specified in the record's GenderCode field. For information about the GenderCode field, see Input on page 771.
Ethnicity	Returns name variations only for the culture specified in the record's Ethnicity field. For information about the Ethnicity field, see Input on page 771.
Romanized	Returns the English romanized version of the name. A romanized name is one that has been converted from a non-Latin script to the Latin script. For example, Achin is the Romanized version of the Korean name 아진.
Native	Returns the name in the native script of the name's culture. For example, a Korean name would be returned in Hangul.
Kana	If you select Native , you can choose to return Japanese names in Kana by selecting this option. Kana is comprised of hiragana and katakana scripts. Note: You must have licensed the Asian Plus Pack database to look up Japanese name variants. For more information, contact your sales executive.
Kanji	If you select Native , you can choose to return Japanese names in Kanji by selecting this option. Kanji is one of the scripts used in the Japanese language. Note: You must have licensed the Asian Plus Pack database to look up Japanese name variants. For more information, contact your sales executive.

Output

Table 145: Name Variant Finder Outputs

Field Name	Format	Description / Valid Values
CandidateGroup	String	Identifies a grouping of an input name and its name variations. Each input name is given a CandidateGroup number. The variations for that input name are given the same CandidateGroup number.
Ethnicity	String	The culture of a name determined by the Core Name and add-on dictionaries. Note: This field was formerly named GenderDeterminationSource.
FirstName	String	The given name of a person.
GenderCode	String	The gender of a name determined by the Core Name and add-on dictionaries. One of the following: M The name is a male name. F The name is a female name. A Ambiguous. The name can be either male or female. U Unknown. The gender of this name is not known.
LastName	String	The last name of a person.
TransactionalRecordType	String	Specifies how the name was used in the matching process. One of the following: Suspect A suspect record is used as input to a query. Candidate A candidate record is a result returned from a query.

Open Name Parser

Open Name Parser breaks down personal and business names and other terms in the name data field into their component parts. These parsed name elements are then subsequently available to other automated operations such as name matching, name standardization, or multiple-record name consolidation.

Open Name Parser does the following:

- Determines the type of a name in order to describe the function that the name performs. Name entity types are divided into two major groups: personal names and business names. Within each of these major groups are subgroups.
- Determines the form of a name in order to understand which syntax the parser should follow for parsing. Personal names usually take on a natural (signature) order or a reverse order. Business names are usually ordered hierarchically.
- Determines and labels the component parts of a name so that the syntactical relationship of each name part to the entire name is identified. The personal name syntax includes prefixes, first, middle, and last name parts, suffixes, and account description terms, among other personal name parts. The business name syntax includes the firm name and suffix terms.
- Parses conjoined personal and business names and either retains them as one record or splits them into multiple records. Examples of conjoined names include "Mr. and Mrs. John Smith" and "Baltimore Gas & Electric dba Constellation Energy".
- Parses output as records or as a list.
- Enables you to use the Open Parser Domain Editor to create new domains that can be used in the Open Name Parser Advanced Options.
- Assigns a parsing score that reflects the degree of confidence that the parsing is correct.

Input

Table 146: Open Name Parser Input

Field Name	Description								
CultureCode	<p>The culture of the input name data. The options are listed below.</p> <table> <tr> <td>Null (empty)</td> <td>Global culture (default).</td> </tr> <tr> <td>de</td> <td>German.</td> </tr> <tr> <td>es</td> <td>Spanish.</td> </tr> <tr> <td>ja</td> <td>Japanese.</td> </tr> </table> <p>Note: If you added your own domain using the Open Parser Domain Editor, the cultures and culture codes for that domain are also valid.</p>	Null (empty)	Global culture (default).	de	German.	es	Spanish.	ja	Japanese.
Null (empty)	Global culture (default).								
de	German.								
es	Spanish.								
ja	Japanese.								

Field Name	Description
Name	The name you want to parse. This field is required.

Options

Open Name Parser options can be configured at the stage level, through any of the Spectrum Technology Platform clients, or at runtime, using dataflow options.

Parsing Options

The following table lists the options that control the parsing of names.

Table 147: Open Name Parser Parsing Options

Option Name	Description
Parse personal names	<p>Specifies whether to parse personal names.</p> <p>Natural The name fields are ordered by Title, First Name, Middle Name, Last Name, and Suffix.</p> <p>Reverse The name fields are ordered by Last Name first.</p> <p>Both The name fields are ordered using a combination of natural and reverse.</p>
Conjoined names	Specifies whether to parse conjoined names.
Split conjoined names into multiple records	<p>Specifies whether to separate names containing more than one individual into multiple records, for example, <i>Bill & Sally Smith</i>.</p> <p>Use a Unique ID Generator stage to create an ID for each of the split records.</p>
Parse business names	Specifies whether to parse business names.
Output results as list	Specifies whether to return the parsed name elements in a list form.

Option Name	Description
Shortcut threshold	<p>Specifies how to balance performance versus quality. A faster performance will result in lower quality output; likewise, higher quality will result in slower performance. When this threshold is met, no other processing will be performed on the record.</p> <p>The default is 100.</p>

Cultures Options

The following table lists the options that control name cultures.

Table 148: Open Name Parser Cultures Options

Option Name	Description
Cultures	<p>Specifies which culture(s) you want to include in the parsing grammar. Global Culture is the default selection.</p> <p>Note: If you added your own domain using the Open Parser Domain Editor, the cultures and culture codes for that domain will appear here as well.</p> <p>Click the Up and Down buttons to set the order in which you want the cultures to run.</p>

Advanced Options

The following table lists the advanced options for name parsing.

Table 149: Open Name Parser Advanced Options

Option	Description
Advanced Options	<p>Use the Domain drop-down to select the appropriate domain for each Name.</p> <p>Click the Up and Down buttons to set the order in which you want the parsers to run. Results will be returned for the first domain that scores higher than the number set in the Shortcut threshold field. If no domain reaches that threshold, results for the domain with the highest score are returned. If multiple domains reach the threshold at the same time, priority goes to the domain that was run first (determined by the order set here) and its results will be returned.</p> <p>Note: If you added your own domain using the Open Parser Domain Editor, that domain will appear here as well.</p>

Configuring Options at Runtime

Open Name Parser options can be configured and passed at runtime if they are exposed as dataflow options. This enables you to override the existing configuration with JSON-formatted name parsing strings. You can also set stage options when calling the job through a process flow or through the job executor command-line tool.

To define Open Name Parser options at runtime:

1. In Enterprise Designer, open a dataflow that uses the Open Name Parser stage.
2. Save and expose that dataflow.
3. Go to `Edit > Dataflow Options`.
4. In the **Map dataflow options to stages** table, expand Open Name Parser and edit options as necessary. Check the box for the option you want to edit, then change the value in the **Default value** drop-down.
5. Optional: Change the name of the options in the **Option label** field.
6. Click **OK** twice.

Output

Table 150: Open Name Parser Output

Field Name	Format	Description
AccountDescription	String	An account description that is part of the name. For example, in "Mary Jones Account # 12345", the account description is "Account#12345".
Names	String	A hierarchical field that contains a list of parsed elements. This field is returned when you check the Output results as list box under Parsing Options.
Fields Related to Names of Companies		
FirmConjunction	String	Indicates that the name of a firm contains a conjunction such as "d/b/a" (doing business as), "o/a" (operating as), and "t/a" (trading as).
FirmName	String	The name of a company. For example, <i>Precisely</i> .
FirmSuffix	String	The corporate suffix. For example, "Co." and "Inc."
IsFirm	String	Indicates that the name is a firm rather than an individual.
Fields Related to Names of Individual People		
Conjunction	String	Indicates that the name contains a conjunction such as "and", "or", or "&".
CultureCode	String	The culture codes contained in the input data.

Field Name	Format	Description
CultureCodeUsedToParse	String	Identifies the culture-specific grammar that was used to parse the data. Null (empty) Global culture (default). de German. es Spanish. ja Japanese. Note: If you added your own domain using the Open Parser Domain Editor, the cultures and culture codes for that domain will appear in this field as well.
FirstName	String	The first name of a person.
GeneralSuffix	String	A person's general/professional suffix. For example, MD or PhD.
IsParsed	String	Indicates whether an output record was parsed. Values are true or false.
IsPersonal	String	Indicates whether the name is an individual rather than a firm. Values are true or false.
IsReverseOrder	String	Indicates whether the input name is in reverse order. Values are true or false.
LastName	String	The last name of a person. Includes the paternal last name.
LeadingData	String	Non-name information that appears before a name.
MaturitySuffix	String	A person's maturity/generational suffix. For example, Jr. or Sr.
MiddleName	String	The middle name of a person.
Name.	String	The personal or firm name that was provided in the input.

Field Name	Format	Description
NameScore	String	Indicates the average score of known and unknown tokens for each name. The value of NameScore will be between 0 and 100, as defined in the parsing grammar. 0 is returned when no matches are returned.
SecondaryLastName	String	In Spanish parsing grammar, the surname of a person's mother.
TitleOfRespect	String	Information that appears before a name, such as "Mr.", "Mrs.", or "Dr."
TrailingData	String	Non-name information that appears after a name.
Fields Related to Conjoined Names		
Conjunction2	String	Indicates that a second, conjoined name contains a conjunction such as "and", "or", or "&".
Conjunction3	String	Indicates that a third, conjoined name contains a conjunction such as "and", "or", or "&".
FirmName2	String	The name of a second, conjoined company. For example, Baltimore Gas & Electric dba Constellation Energy.
FirmSuffix2	String	The suffix of a second, conjoined company.
FirstName2	String	The first name of a second, conjoined name.
FirstName3	String	The first name of a third, conjoined name.
GeneralSuffix2	String	The general/professional suffix for a second, conjoined name. For example, MD or PhD.

Field Name	Format	Description
GeneralSuffix3	String	The general/professional suffix for a third, conjoined name. For example, MD or PhD.
IsConjoined	String	Indicates that the input name is conjoined. An example of a conjoined name is "John and Jane Smith."
LastName2	String	The last name of a second, conjoined name.
LastName3	String	The last name of a third, conjoined name.
MaturitySuffix2	String	The maturity/generational suffix for a second, conjoined name. For example, Jr. or Sr.
MaturitySuffix3	String	The maturity/generational suffix for a third, conjoined name. For example, Jr. or Sr.
MiddleName2	String	The middle name of a second, conjoined name.
MiddleName3	String	The middle name of a third, conjoined name.
TitleOfRespect2	String	Information that appears before a second, conjoined name, such as "Mr.", "Mrs.", or "Dr."
TitleOfRespect3	String	Information that appears before a third, conjoined name, such as "Mr.", "Mrs.", or "Dr."

Open Name Parser Summary Report

The Open Name Parser Summary Report lists summary statistics about the job, such as the total number of input records and the total number of records that contained no name data, as well as several parsing statistics. For instructions on how to use reports, see the *Spectrum Technology Platform Dataflow Designer's Guide*.

General Results

- **Total number of input records**—The number of records in the input file.
- **Total number of records that contained no name data**—The number of records in the input file that did not contain name data to be parsed.
- **Total number of names parsed out**—The number of names in the input file that were parsed.
- **Total Records**—The total number of records processed.
- **Lowest name parsing score**—The lowest parsing score given to any name in the input file.
- **Highest name parsing score**—The highest parsing score given to any name in the input file.
- **Average name parsing score**—The average parsing score given among all parsed names in the input file.

Personal Name Parsing Results

- **Number of personal name records written**—The number of personal names in the input file.
- **Number of names parsed from conjoined names**—The number of parsed names from records that contained conjoined names. For example, if your input file had five records with two conjoined names and seven records with three conjoined names, this value for this field would be 31, as expressed in this equation: $(5 \times 2) + (7 \times 3)$.
- **Records with 2 conjoined names**—The number of input records containing two conjoined names.
- **Records with 3 conjoined names**—The number of input records containing three conjoined names.
- **Number of names with title of respect present**—The number of parsed names containing a title of respect.
- **Number of names with maturity suffix present**—The number of parsed names containing a maturity suffix.
- **Number of names with general suffix present**—The number of parsed names containing a general suffix.
- **Number of names that contained account descriptions**—The number of parsed names containing an account description.
- **Total Reverse Order Names**—The number of parsed names in the reverse order, resulting in the output field isReversed as "True".

Business Name Parsing Results

- **Number of business name records written**—The number of business names in the input file.
- **Number of names with firm suffix present**—The number of parsed names containing a firm suffix.
- **Number of names that contained account descriptions**—The number of input records containing an account description.
- **Total DBA Records**—The number of input records containing Doing Business As (DBA) conjunctions, resulting in both output fields isPersonal and isFirm as "True".

Flow Output

Output stages are also known as Sinks and Write-to stages.

To define the output from a flow, use a "sink" stage. A sink is the last stage in a flow. It defines what to do with the output from the flow. A sink can also perform other actions at the end of a flow, such as executing a program.

Note: This version of Spectrum Flow Designer allows you to use input files and to write to output files that are on the Spectrum Server location, only. At this time, Spectrum Flow Designer does not support *local* input and output files.

Output from a Job

Output from a job can be written to a file or a database. Spectrum Technology Platform has the ability to write data to many file formats and database types. The types of the ability to write data to many file formats and database types. The types of sinks you can write to depend on which modules you have licensed. See the solution guide for your modules available at docs.precisely.com.

Output from a Service

Output data from a service is defined in an Output stage. This stage defines the fields that the service will return in response to a web service request or an API call.

Other output stages

Additional output Sink stages include:

- Terminate Job – Add to a flow to stop a job at a specified point or for a specific reason.
- Write to Null – Counts records, then discards the records that you choose to not keep according to flow processing criteria you specify. Use this stage if there are records that you do not want to preserve after the dataflow finishes.
- Write to XML – Sends output to an XML-format output file that can be consumed by other processes or flows.

These additional Sink stages are not configurable.

Defining Service Output

The Output stage defines the output fields that the service or subflow returns. Follow the steps below to define the service output.

1. Double-click the Output icon on the canvas. The **Output Options** dialog box appears. When you open the **Output Options** dialog box for the first time, a list of fields defined in the Input is displayed.
2. To add a new field to the field list, click **Add**. The **Add Custom Field** dialog box appears. You can also modify or delete a custom field.
3. Click **Add** again.
4. Type the field name in the text box.
5. Select the **Data type** and press **OK**. These data types are supported:

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.

bytearray An array (list) of bytes.

Note: Bytearray is not supported as an input for a REST service.

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 $(2-2^{-52}) \times 2^{1023}$. 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 $(2-2^{-23}) \times 2^{127}$. In E notation, the range of values is -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).

list Strictly speaking, a list is not a data type. However, when a field contains hierarchical data, it is treated as a "list" field. In Spectrum Technology Platform a list is a collection of data consisting of multiple values. For example, a field

Names may contain a list of name values. This may be represented in an XML structure as:

```
<Names>
  <Name>John Smith</Name>
  <Name>Ann Fowler</Name>
</Names>
```

It is important to note that the Spectrum Technology Platform list data type different from the XML schema list data type in that the XML list data type is a simple data type consisting of multiple values, whereas the Spectrum Technology Platform list data type is similar to an XML complex data type.

- 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.

You can also add a new, user-defined data type if necessary, and that new type can be a list of any defined data type. For example, you could define a list of names (string), or a new data type of addresses that includes AddressLine1 (string), City (string), StateProvince (string) and PostalCode (string). After you create the field, you can view the data type by accessing the Input Options dialog and pressing the button in the Data Type column. The **Data Type Details** dialog box will appear, showing the structure of the field.

6. Click **OK** again.
7. Click the check box next to **Expose** to select the check box of all fields in the field list. Selecting a field in the field list exposes it to the dataflow for stage operations. Click the check box again to clear the check box for all fields in the list. Clearing the check box of one or more fields in the field list and clicking **OK** deletes the field from the field list.

Note: If you define hierarchical data in the input fields, you will not be able to import data or view the data vertically.

8. Click **OK** to return to the canvas.

Defining A Web Service Data Type

The **Data type name** field allows you to control the WSDL (SOAP) and WADL (REST) interfaces for the service you are creating. The name of the Rows element is determined by the name you give this stage in the service, and the name of the Row element is determined by the text you enter here.

Note: For WSDL, both requests and responses are affected, but for WADL only responses are affected.

Prior to naming this stage and entering text in this field, your code might look like this:

```
<Rows>
  <Row>
    <FirstName>John</FirstName>
    <LastName>Doe</LastName>
  </Row>
  <Row>
    <FirstName>Jane</FirstName>
  <LastName>Doe</LastName>
  </Row>
</Rows>
```

After naming this stage and entering text in this field, your code might look like this:

```
<Names>
  <Name>
    <FirstName>John</FirstName>
    <LastName>Doe</LastName>
  </Name>
  <Name>
    <FirstName>Jane</FirstName>
    <LastName>Doe</LastName>
  </Name>
</Names>
```

Running an External Program

An Execute Program stage invokes an executable entity, such as a program or command line command, when it receives a record. To use an Execute Program stage in your flow:

Options

Option	Description
Command-line	The executable name and arguments (if applicable). The arguments can be data available in the flow. To access that data, click the [...] (Browse) button. You can select from the following three contexts: Current Job ID, Current Job Name, or Current User Name. You can also select from the available fields. For example, JobStatus and JobComment.

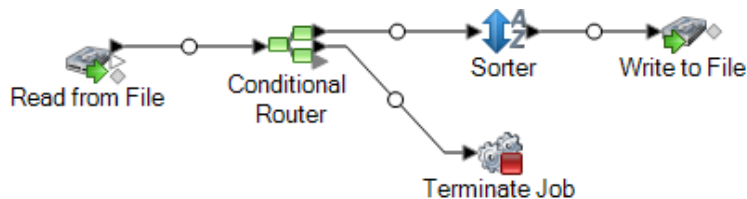
Option	Description
Timeout	<p>Specifies whether to cancel the job if the command does not respond within a given amount of time:</p> <p>No timeout Do not cancel the execution if the command fails to respond.</p> <p>Timeout in milliseconds Cancels the execution attempt if the command does not respond in the specified number of milliseconds.</p>
Environment Variables	<p>Optional. Specifies environment variables to use when executing the command. To add an environment variable click Add.</p> <p>Enter the appropriate key word in the Key field, such as "JAVA_HOME".</p> <p>Enter the appropriate value in the Value field, such as C:\Java\jre7. Alternatively, you can select a field from the Field List dialog box by clicking the [...] (Browse) button. You can select from one of these contexts: Current Job ID, Current Job Name, or Current User Name. You can also select from the available fields, such as JobStatus and JobComment.</p>

Terminating a Job Based on a Condition

The Terminate Job stage is used in combination with Conditional Router to end a job if certain criteria are found within a record. If a record is sent to Terminate Job, the job ends.

Note: Terminate Job is not available in services or subflows.

To use Terminate Job, add a Conditional Router and a Terminate Job stage to your flow. Then connect the stages and configure the Conditional Router. The Conditional Router should be configured to contain the criteria you want to trigger job termination. When a record is found that meets the criteria, it is passed to Terminate Job and the job terminates, producing a message that says "Job terminated by stage: <stage label>." The completed flow should look something like this:



Discarding records - Write to Null

The Write to Null stage counts records then discards them. Use this stage if there are records that you do not want to preserve after the flow runs.

Embedded flows

An embedded flow reduces the number of stages displayed on the canvas at one time by grouping stages together. The grouped stages then appear as a single stage. You can use embedded flows to:

- Simplify the layout of complex flows by grouping stages together and having them represented as one stage on the canvas.
- Process records in groups using the iteration feature.
- Use a value in a field to set stage options in the embedded flow.

You can add an unlimited number of embedded flows to a flow, and you can put embedded flows within embedded flows.

Differences between embedded flows and subflows

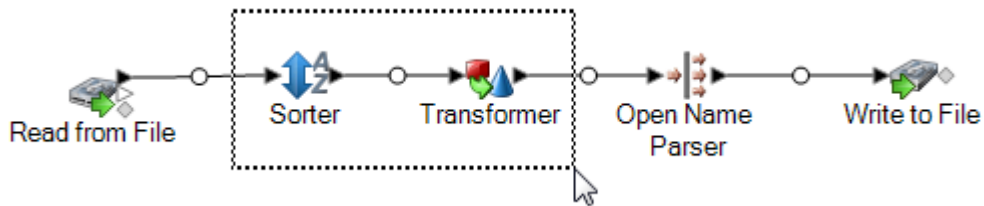
There are two major differences between an embedded flow and a subflow. First, iteration processing is only available in embedded flows. Iteration allows you to process groups of records together for purposes such as aggregating records for processing or setting stage options based on field values. The second difference between an embedded flow and subflow is that an embedded flow cannot be used in more than one flow. If you want to reuse a portion of a flow in multiple flows, create a subflow instead of an embedded flow. Embedded flows can be converted into subflows if you decide you want to reuse an embedded flow in other flows, but when you convert an embedded flow to a subflow its iterations options are removed.

Grouping stages into an embedded flow

An embedded flow groups stages together into a single stage, allowing you to simplify the layout of complex flows and set processing options using field values.

1. In your flow, add the stages that you want to convert to an embedded flow.
2. Select the stages you want to convert to an embedded flow by clicking and dragging a box around the stages.

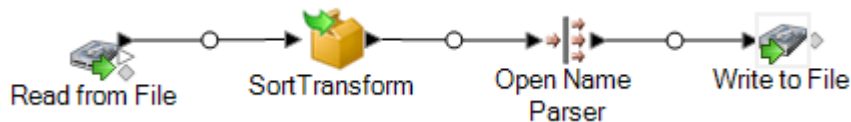
For example, this would select the Sorter and Transformer stages to convert to an embedded flow.



Note: Report stages cannot be included in an embedded flow.

3. Right-click one of the selected stages and select **Group into Embedded Dataflow**
4. Enter a name for the embedded flow. The name will be used as the label for the embedded flow stage on the canvas.
5. Click **OK**.

The stages you selected are now grouped into an embedded flow. This example shows an embedded flow named SortTransform.

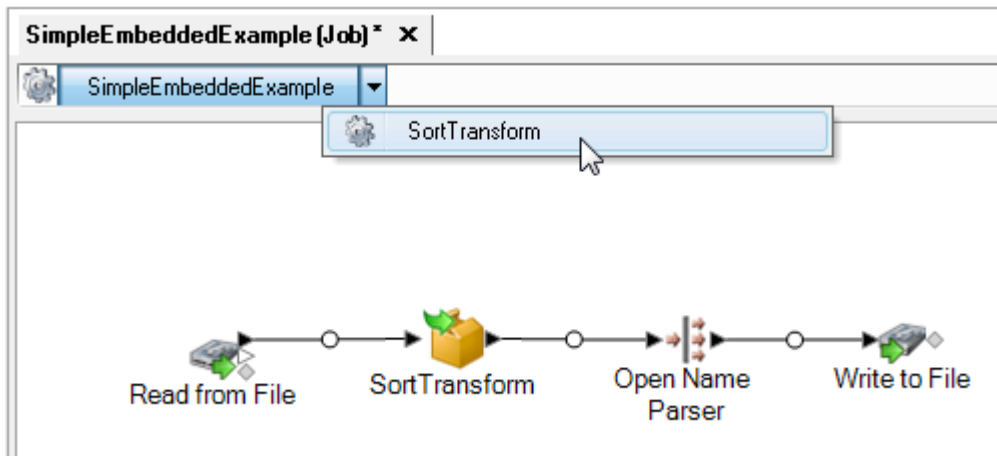


Editing an embedded flow

An embedded flow groups stages together into a single stage, allowing you to simplify the layout of complex flows and set processing options using field values.

1. Right-click the icon and select **Edit This Embedded Dataflow**.

Tip: Alternatively, you can use the breadcrumb links at the top of the flow to open embedded flows. For example, this shows how to open an embedded flow named SortTransform:



When you open an embedded flow you will see an Input stage and an Output stage. These represent the input to the embedded flow from the parent flow, and the output to the parent flow from the embedded flow.

2. Modify the embedded flow as needed.

Any changes you make to an embedded flow are saved the next time you save the parent flow.

Using iteration with an embedded flow

Iteration settings specify how an embedded flow should process incoming records. By default, an embedded flow processes each record individually just as any other stage in the flow would. But if you use iteration, you can process groups of records together, which can be useful for things like performing comparisons or calculations based on groups of records rather than the entire set of input data. You can also use iteration to set stage options based on the data in each record.

There are two kinds of iteration: per-record iteration and per-group iteration. In per-record iteration, an embedded flow process one record at a time and the result is sent along to the next stage following the embedded flow. Per-record iteration is useful if you want to set stage options on a record-by-record basis using field values.

In per-group iteration, records are grouped by a key field and the embedded flow processes each group. All the records in a group are processed in one iteration, then the group is written to the next stage following the embedded subflow. Use per-group iteration to perform processing on groups of related records, as well as to set stage options to use when processing the group of records. For example, you might want to group records by customer ID so that you can perform an analysis of each customer's records, perhaps to determine which store each customer visits most often.

You should consider the impact on performance when using iteration. Each time a new iteration starts, there is some overhead during the initialization of the embedded flow, and this overhead can be significant, especially if you have embedded flows within other embedded flows. For example, if the an embedded flow iterates 1,000 times and it contains within it another embedded flow that also

iterates 1,000 times, the total number of iterations would be 1,000,000. Using per-record iteration has a more significant impact on performance since each record kicks off a new iteration.

1. Create an embedded flow containing the stage or stages that you use for iteration.

Note: There are some limitations to what can be included in embedded flows that have iteration enabled:

- The Stream Combiner stage cannot be the first stage in an embedded flow that has iteration enabled.
- The embedded flow cannot contain a sink that writes to a file located on the client. Sinks inside an embedded flow must write to a file on the Spectrum Technology Platform server or on a file server.

2. Double-click the embedded flow icon.
3. Check the **Enable iteration** check box.
4. If there is more than one input channel connected to the embedded flow, use the **Port** field to choose the port whose records you want to use to drive iteration.

For example, say you have two input ports, A and B, and you choose to iterate each time a key field changes. If you choose to use port B for iteration, the embedded flow will start a new iteration each time a key field in the records from port B changes. All the records from the other port, port A, will be read into the embedded flow, cached, and used for each iteration.

5. Select the type of iteration you want to perform.

Iterate each time a key field changes In this type of iteration, the embedded flow processes groups of records that have the same value in one or more fields. When the embedded flow finishes processing the group of records, the embedded flow resets and a new group of records is processed. Use this type of iteration to create embedded flows that process groups of records and groups each output record group separately.

Tip: If you choose this type of iteration, you can improve performance by placing a Sorter stage in front of the embedded flow and sorting the records by the key field.

Iterate per record In this type of iteration, the embedded flow processes one record at a time. Every time one record completes the embedded flow processing, the result is sent to the output and a new record is processed. Embedded flows that iterate for the record handle each record as a new flow execution.

6. If you choose **Iterate each time a key field changes**, check the box **Ignore case when comparing values** if you want to ignore differences in case when evaluating key field values to determine record groups.
7. Specify one or more key fields.
 - a) Click **Add**.
 - b) Choose the field you want to use as a key field.

- c) If you want to use the field's value to set a stage option within the embedded flow, specify the name of the option you want to set.
- d) Click **OK**.
- e) Add additional key fields if needed.

If you have more than one key field and you chose the option **Iterate each time a key field changes**, records must contain the same value in all key fields to be grouped together.

Ungrouping an embedded flow

When you ungroup an embedded flow, the stages from the flow are placed in the parent flow at the location of the embedded flow. This effectively removes any iteration settings you may have configured for the embedded flow.

To ungroup the stages from an embedded flow, right-click the embedded flow and select **Ungroup this Embedded Dataflow**.

Converting an embedded flow to a subflow

If you want to reuse an embedded flow in another flow, you must convert the embedded flow to a subflow. This is because embedded flows cannot be used in other flows. Once you convert an embedded flow to a subflow, it can be used like any other subflow.

Note: If the embedded flow has iteration enabled, iteration settings will be removed when it is converted to a subflow. Subflows do not support iteration.

1. Open the flow containing the embedded flow that you want to convert to a subflow.
2. Right-click the embedded flow and select **Convert Stage to Subflow**.
3. Enter a name for the new subflow and click **OK**.

The embedded flow is converted to a subflow and available to use in other flows.

Reports

Spectrum Technology Platform provides reporting capabilities for jobs. You can use standard reports that come with some Spectrum processes or you can design your own reports. When a report is included in a flow the entire flow runs, and after completion the report stages in the flow are run and the reports are saved in the format you choose, for example PDF.

Adding a standard report to a job

A standard report is a configured report that is included with a Spectrum Technology Platform module. For example, Spectrum Spatial includes the Point In Polygon Summary report, which summarizes the results of point in polygon calculations, such as the number of polygon matches, the database used for the job, and other information.

This procedure describes how to add a standard report to a job.

1. In Spectrum Enterprise Designer, on the left side of the window under Palette, click **Reports**. A list of available reports appears.
2. Drag the report you want to the canvas. You do not need to connect the report icon to anything.
3. Double-click the report.
4. Select the stages that you want to contribute to the report.
5. Click the **Parameters** tab.
6. Clear the **Use default reporting options** check box and select the appropriate output format if you wish to specify a format other than PDF (such as html or txt).

Setting report options for a job

Reports provide summary information about a job, such as the number of records processed, and the settings used for the job. Report options specify how to handle the reports generated by a job, such as the output format and archiving options. Default values for report options are specified in Spectrum Management Console but you can override the default options for a job in Spectrum Enterprise Designer.

This procedure describes how to specify report options for a job.

1. Open the job in Spectrum Enterprise Designer and go to **Edit > Job Options**.
2. Click the **Reporting** tab.
3. Clear the **Use global reporting options** check box.
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: <table> <tr> <td>Job ID</td> <td>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.</td> </tr> <tr> <td>Stage</td> <td>The name of the stage that contributed data to the report, as specified in the report stage in Enterprise Designer.</td> </tr> <tr> <td>Date</td> <td>The day, month, and year that the report was created.</td> </tr> </table>	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.
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.						

7. Click **OK**.

When you run your job, the Execution History will contain a column that shows if there are any reports that are associated with the job. An empty icon indicates no reports, one document icon indicates one report, and multiple documents icons indicate multiple reports. You can use the Job Detail to view, save, or print the report.

Note: To delete a report, right-click the report icon on the canvas and select **Delete**.

Viewing reports

To view reports, first run the job then do one of these steps:

- In Spectrum Enterprise Designer, the **Execution Details** window will appear when you run your job. Select the report you want to view.
- In Spectrum Management Console, in the Execution node, click **History** then select the job whose reports you want to view, then click **Details**.

Note: If the Java Virtual Machine does not have the required fonts, the downloaded reports will be empty files and the error logs will indicate that the fonts are not available to the JVM. To install the missing font files (.ttf), follow the instructions here:

<https://support.azul.com/hc/en-us/articles/360034030692-Using-Fonts-with-OpenJDK-Zulu>.

Using custom reports

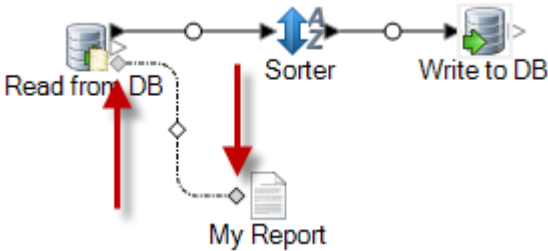
Important: When generating custom reports, ensure that the stages that reference the files used by the reports reside on the Spectrum server or another file server. The Spectrum server cannot access local files to generate custom reports.

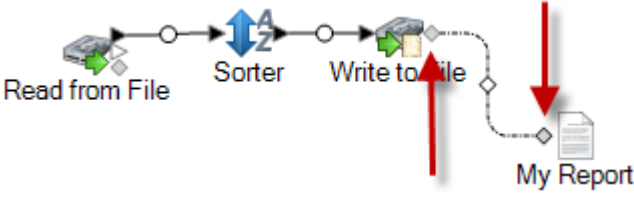
Spectrum Technology Platform modules come with reports that are useful for basic reporting. However, if you have report requirements that are not met by the standard reports, you can create your own custom reports and include them in your flow.

1. Create the report template using the report design tool of your choice. Your design tool must be able to export the report in the JasperReports format (*.jrxml).
2. Copy your *.jrxml file to the `server\import` folder on the Spectrum Technology Platform server.

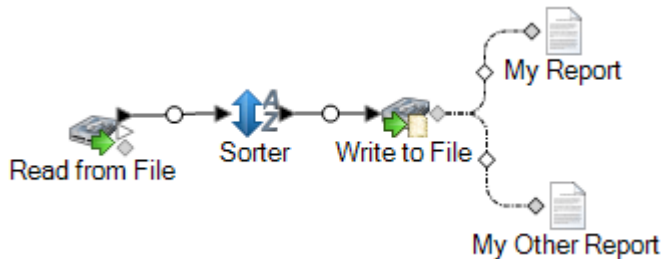
Within a few seconds, the report template will be imported into the system and made available in Spectrum Enterprise Designer.

3. In Spectrum Enterprise Designer, open the job to which you want to add your custom report.
4. On the left side of the window, under Palette, click **Reports**.
5. Drag your custom report to the canvas.
6. Do one of these tasks to specify the data source for the report:

Option	Description
<p>To report on the flow's input</p>	<p>Connect the report to the source stage you want to report on using the gray diamond-shaped report port as shown here:</p>  <p>The report will be based on the flow's input data and will not reflect any of the processing that occurs in the flow.</p>

Option	Description
To report on the flow's output	<p>Connect the report to the sink stage you want to report on using the gray diamond-shaped report port as shown here:</p>  <p>The report will be based on the flow's output data and will reflect the flow's effect on the data.</p>
To use a query embedded in the report template	<p>If the report template contains an embedded SQL query in the <code><queryString></code> element of the JRXML file, double-click the report icon and check the Use embedded query box, then select the database connection to use for the query.</p> <p>Note: If you need to define a database connection, open the Spectrum Management Console and go to Resources, then Connections.</p>

You can connect multiple reports to a source or sink, as shown here:



7. If the report contains user-defined parameters:
 - a) Double-click the report icon on the canvas.
 - b) On the **Parameters** tab, specify the values you want to use for the report's user-defined parameters.
8. Optional: If necessary, right-click on the channel and map the fields from the source or sink to the fields in the report.

Performance Considerations

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 to 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.

Stage Runtime Performance Options

Runtime performance options control how individual stages in a flow are run and provide settings you can use to improve the performance of your flow. The settings available to you depend on how your Spectrum Technology Platform environment has been configured.

- The **Local** option is the default setting in which stages run on the local Spectrum Technology Platform server and use one runtime instance. The runtime instances setting can be increased, thereby utilizing parallel processing and improving performance.
- The **Distributed** option is typically used in a clustered environment which involves installing a load balancer and multiple Spectrum Technology Platform servers.
- The **Remote** option can be used if your environment consists of multiple Spectrum Technology Platform servers but is not configured for distributed processing. This option allows you to have a stage's processing performed by another server.

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

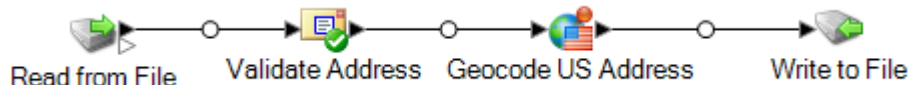
Spectrum databases contain reference data used by certain stages, such as postal data used to validate addresses, or geocoding data used to geocode addresses. These databases can be configured to accept multiple concurrent requests from the stages or services that use them, thereby improving the performance 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](#).
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 834.

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.

Distributed Processing

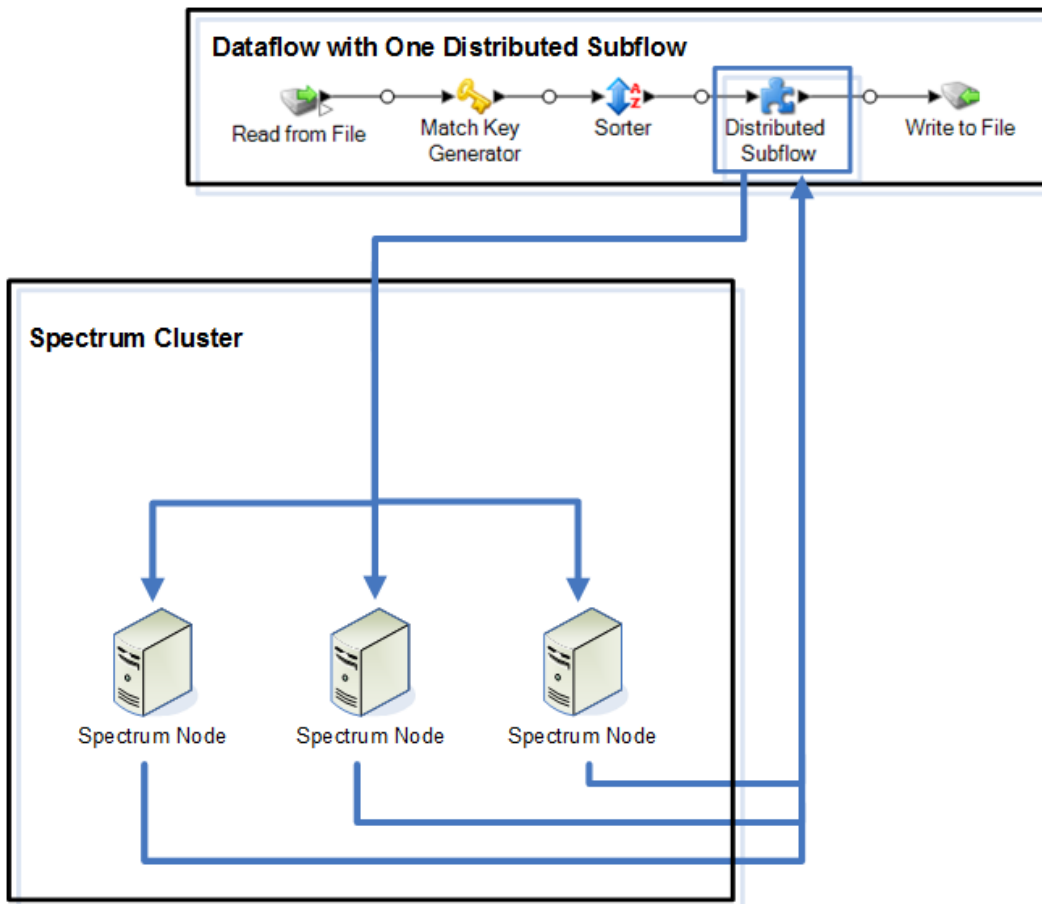
If you have a very complex job, or you are processing a very large data set such as one containing millions of records, you may be able to improve flow performance by distributing the processing of the flow to multiple instances of the Spectrum Technology Platform server on one or more physical servers.

The most scalable solution for distributed processing is to install Spectrum Technology Platform in a cluster. See the *Installation Guide* for instructions on installing and configuring a cluster.

Note: While it is also possible to use distributed processing on a single Spectrum Technology Platform server, the following information describes using distributed processing in a cluster. If you are using a single server, distributed subflow processing is broken up into microbatches and processed by the one server instead of by the cluster.

Once your clustered environment is set up, you can build distributed processing into a flow by creating subflows for the parts of the flow that you want to distributed to multiple servers. Spectrum Technology Platform manages the distribution of processing automatically after you specify just a few configuration options for the subflow.

Distributed processing looks like this:



As records are read into the subflow, the data is grouped into batches. These batches are then written to the cluster and automatically distributed to the a node in the cluster which processes the batch. This processing is called a microflow. A subflow may be configured to allow multiple microflows to be processed simultaneously, potentially improving performance of the flow. When the distributed instance is finished processing a microflow, it sends the output back into the parent flow.

The more Spectrum Technology Platform nodes you have the more microflows can be processed simultaneously, allowing you to scale your environment as needed to obtain the performance you require.

Once set up, a clustered environment is easy to maintain since all nodes in the cluster automatically synchronize their configuration, which means the settings you apply through Spectrum Management Console and the flows you design in Spectrum Enterprise Designer are available to all instances automatically.

Designing a flow for distributed processing

Distributed processing takes parts of your flow and distributes the processing of those parts to a cluster of Spectrum Technology Platform servers. For example, your flow may perform geocoding, and you might want to distribute the geocoding processing among several Spectrum Technology Platform nodes in a cluster to improve performance.

1. Decide which stages of your flow you want to distribute, then create a subflow containing the stages that you want to distribute.

Do not use these stages in a subflow that will be used for distributed processing:

- Sorter
- Unique ID Generator
- Record Joiner
- Interflow Match

These sets of stages must be used together in a subflow for distributed processing:

- Matching stages (Intraflow Match and Transactional Match) and consolidation stages (Filter, Best of Breed and Duplicate Synchronization).
- Aggregator and Splitter

Do not include other subflows within the subflow (nested subflows).

Note these exceptions if you will be performing matching operations in a subflow used for distributed processing:

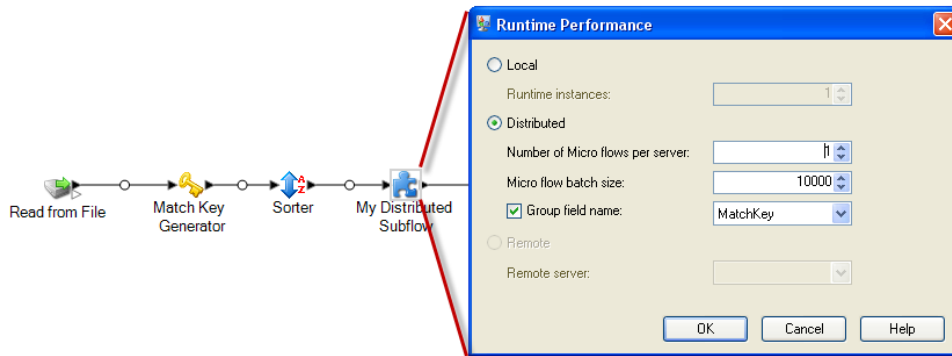
- Sorting must be done in the job and not in the subflow. You must turn sort off in the stage and put the sort at job level.
- Match Analysis is not supported in a distributed subflow
- Collection numbers will be reused within a microflow batch group

Using a Write Exception stage in a subflow may produce unexpected results. Instead, you should add this stage to your flow at the job level.

2. Once you have created your subflow for the portion of the flow you want to distribute, add the subflow to the parent flow and connect it to an upstream and downstream stage. Subflows used for distributed processing may have only one input port.
3. Right-click the subflow and select **Options**.
4. Select **Distributed**.
5. Enter the number of microflows to be sent to each server.
6. Enter the number of records that should be in each microflow batch.
7. Optional: (Optional) Check **Group field name** and select the name of the field by which the microflow batches should be grouped.

If you provide a group field, your batch sizes could be greater than the number you specified in the **Micro flow batch size** field because a group will not be split across multiple batches. For example, if you specify a batch size of 100, but you have 108 records within the same group, that batch will include 108 records. Similarly, if you specify a batch size of 100, and a new group of 28 records with the same ID starts at record 80, you will have 108 records in that batch.

This example shows a flow where a subflow named My Distributed Subflow has been configured to run in distributed mode:



Running a Stage on a Remote Server

If your system administrator has enabled remote servers in Spectrum Management Console, you can have stages in your flow run on a remote server. Using remote servers can improve performance by spreading flow processing among multiple Spectrum Technology Platform servers.

Your system administrator may have already designated certain stages to run on a remote server. If a stage is already routed to a remote server, you will see a red star in the top-left corner of the stage icon on the canvas in Spectrum Enterprise Designer.

This procedure describes how to configure remote processing for a stage in a flow.

1. Open the flow in Spectrum Enterprise Designer.
2. Double-click the stage you want to route to a remote server.
3. Click **Runtime**.

The **Runtime Performance** dialog appears.

4. Click **Remote** and select the remote server to which you wish to route the process for this stage.
5. Click **OK**.

Troubleshooting Remote Server Errors

This section discusses possible errors you may experience when using remote servers.

Module Not Licensed

The remote server must have the license for both the module and the mode of execution you are trying to run, either batch or real-time. The license on the remote server may be different from the license on the local server. Log in to the remote server using Spectrum Management Console and verify that the correct license is installed. You must log in with an account that has administrative privileges in order to view license information.

Remote Server Not Available

If the remote server is not running or is not reachable for any other reason, the remote services will become unavailable in Spectrum Enterprise Designer and Spectrum Management Console. You will see a yellow hazard icon in the status bar at the bottom of the screen:



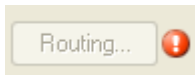
Click this icon to see an error message that describes which remote servers are not available.

In addition, in Spectrum Enterprise Designer any stages that use a remote stage will be replaced with an icon showing you the stage is no longer available:



Routing Has Changed

If you delete or undeploy a service that is installed both locally and remotely and has been routed through a remote server, and click that service within Spectrum Management Console, you will see a routing change indicator (a blinking exclamation point) next to the routing button on the Options tab for that service. This indicator means the routing has changed for that service.



Optimizing 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
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 $N \times (N-1) \div 2$. 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 $N \times (N-1)$ comparisons. If all records match, the number of comparisons is $N-1$. If no records match the number of

comparisons is $N \times (N-1)$. 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 $N \times W$ 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.

Flow Versions

The Versions feature in Spectrum Enterprise Designer allows you to keep a revision history of your flows. You can view previous versions of a flow, expose older versions for execution, and keep a history of your changes in case you ever need to revert to a previous version of a flow.

Saving a Flow Version

There are two ways to save a version of your flow in Spectrum Enterprise Designer:

- Expose your flow. Each time you expose a flow, either by selecting **File > Expose/Unexpose and save** or by clicking the light bulb in the tool bar, Spectrum Enterprise Designer automatically saves a version of the flow.
- Manually save a version in the **Versions** pane in Spectrum Enterprise Designer.

Note: A flow version is not created when you simply save a flow.

This procedure describes how to manually save a version in the **Versions** pane of Spectrum Enterprise Designer.

1. In Spectrum Enterprise Designer, open the flow.
2. If the **Versions** pane is not visible, select **View > Versions**

3. Make sure that the latest saved version is selected in the **Versions** list. This is the version at the top of the list.
4. Click the green plus icon in the **Versions** pane.

A new version of the flow is saved and added to the **Versions** pane.

Viewing a Flow Version

You can view a previous version of a flow. This allows you to see how a flow was designed in the past before more recent changes were made. Previous versions can only be viewed, not modified. In order to modify a previous version it must first be promoted to the latest saved version.

1. In Spectrum Enterprise Designer, open the flow.
2. If the **Versions** pane is not visible, select **View > Versions**
3. Select the version that you want to view.

The selected version is displayed on the flow canvas.

Editing a Flow Version

You can edit a previous version of a flow by promoting it to the latest-saved version. Promoting a flow version moves it to the latest-saved version, making it available for editing.

Note: Before performing this procedure, note that the existing latest-saved version will be overwritten by the version you promote and edit. If you want to preserve a copy of the existing latest-saved version, save it as a version before promoting the older version.

1. In Spectrum Enterprise Designer, open the flow.
2. If the **Versions** pane is not visible, select **View > Versions**
3. Select the version that you want to edit.
4. Click the promote icon.



The selected version is promoted to the latest-saved version. You can now edit the flow.

Editing Version Properties

When you save a flow version, it is given a default version number. You can modify the version number and add comments to document the version's changes or purpose.

1. In Spectrum Enterprise Designer, open the flow.
2. If the **Versions** pane is not visible, select **View > Versions**
3. Select the version that you want to modify.
4. Click the properties icon:



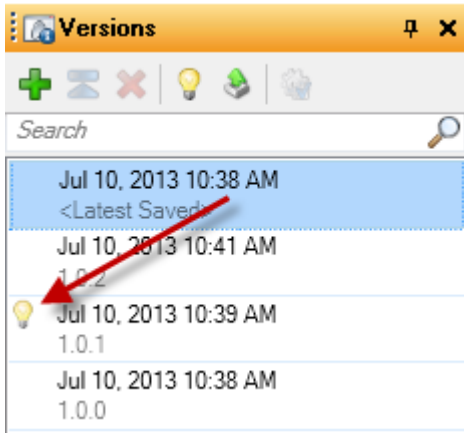
5. In the **Name** field, enter a name for the version. You can use version numbers or any meaningful name. The name can be anything you choose.
6. In the **Comment** field, you can enter a longer comment that describes in more detail the purpose of the version of the changes you made. Adding a comment is optional.
7. Click **OK**.

Exposing a Version

If you have saved multiple versions of a flow you can choose which version to expose for execution.

1. In Spectrum Enterprise Designer, open the flow.
2. If the **Versions** pane is not visible, select **View > Versions**
3. In the **Versions** pane, select the version of the flow that you want to expose.
4. Select **File > Expose/Unexpose and Save**

The selected version is now exposed and available for execution. The version with the light bulb next to it is the version that is exposed, as shown here:



When a flow is exposed the light bulb button in the Spectrum Enterprise Designer tool bar indicates that the flow is exposed as shown here:



The light bulb indicates that the flow is exposed even if you are viewing a version other than the exposed version. If you click the light bulb while viewing an unexposed version it will switch the exposed version to the version you are currently viewing. If you click the light bulb while viewing the exposed version, it will unexpose the flow.


3 - Inspecting and Testing

In this section

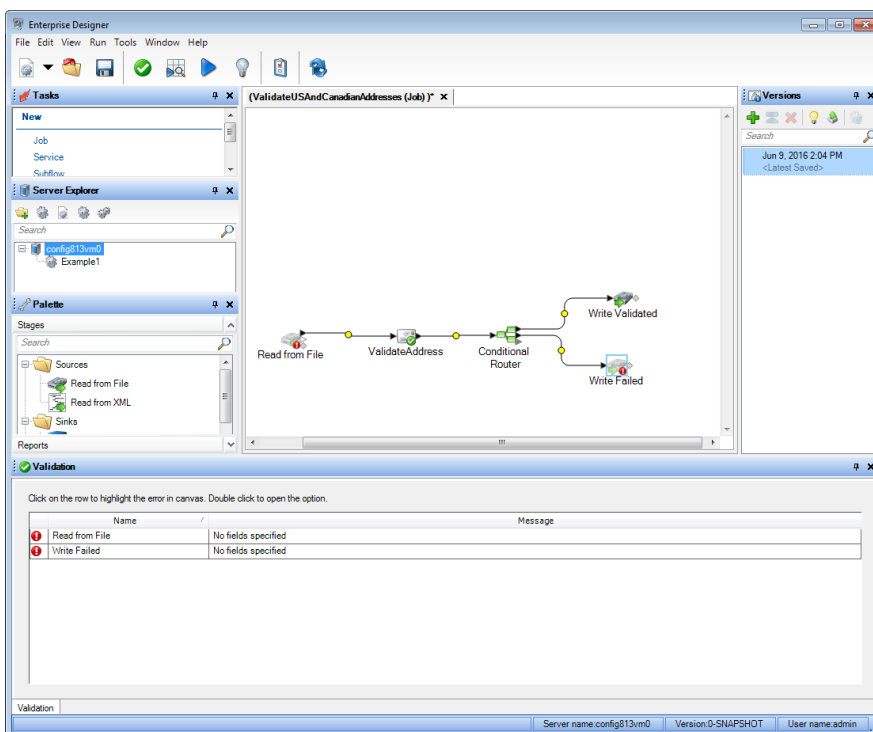
Checking a Flow for Errors.....	815
Inspecting a flow.....	815
Testing a service with Spectrum Management Console.....	819



Checking a Flow for Errors

Dataflow Designer automatically checks a flow for errors when you run a flow, run inspection, expose a flow, or save an exposed flow. You can also check for errors by clicking the validation button .

When an error is found, the **Validation** pane appears at the bottom of the Spectrum Enterprise Designer window. Click an error to highlight the error on the canvas. Double-click an error to open the options window of the item containing the error.



Inspecting a flow

Use the inspection tool to view the effect of your dataflow on the input data at different points in the dataflow. Inspection enables you to confirm that the dataflow is having the desired effect on your data, isolate problems, or identify records that contain defects.

1. Specify the data to use for inspection.

The data should be representative of actual data, or, if you are troubleshooting a specific issue, should be the data that causes the issue you are troubleshooting. There are two ways to specify the data to use for inspection, depending on whether you are inspecting a service or a job.

Scenario	Description
<p>To specify inspection data for a job</p>	<p>When inspecting a job, the data used for inspection is the data specified in the source stage. The inspection tool can process a maximum of 50 records, which by default is the first 50 records in the input file or database. If you want to use data that starts somewhere other than the first record, double-click the Read From File stage and complete the Starting record field in the Runtime tab.</p>
<p>To specify inspection data for a service</p>	<p>Service dataflows use an Input stage to define the input to the dataflow. Because an Input stage does not have access to data when you are editing the dataflow, you must define inspection data in the Input stage on the Inspection Data tab. You can specify a maximum of 50 records.</p> <p>There are a few ways you can enter inspection data in an Input stage.</p> <ul style="list-style-type: none"> • If you want to use just a few records for inspection, you can manually type in the data. <p>Tip: If want to save the inspection data you enter to use again in another stage, you can export the inspection data to a text file by clicking Export Data.</p> <ul style="list-style-type: none"> • If you have data in a CSV or TXT file, you can import the data by clicking Import Data. The data must use one of these delimiters: <ul style="list-style-type: none"> • \t • • , • ; • You can copy delimited data from another application and paste it into the inspection data editor. <p>The Inspection Input tab indicates pass-through data by enclosing the field name in parentheses, as shown here:</p>

Note: Certain field types have restrictions when used for inspection:

- Double and float fields must contain numeric data only. The field may have up to 16 digits and 6 decimal places. Exponential notation is not supported in inspection.
- Integer and long fields must contain numeric data only.

2. Indicate the points in the dataflow where you want to view data.

Scenario	Description
<p>To add an inspection point to a channel</p>	<p>Right-click to the left of the Rename node on a channel and select Add Inspection Point.</p> <p>A point is added to the job:</p> <hr/>
<p>To compare records at two points in a dataflow</p>	<p>Add two inspection points at the points in the dataflow that you want to compare:</p> <p>Tip: If you are using inspection to identify a problem, inspect outer points on the dataflow first then move inward to narrow down where a problem may be.</p> <hr/>
<p>To inspect a subflow embedded in a job or service</p>	<p>Right-click the subflow stage and select Inspect this Dataflow:</p> <p>The input data (in a job) or the inspection data (in a service) is automatically passed to the subflow, so there is no need to enter inspection data in the subflow's Input stage.</p> <p>Note: When you inspect a subflow, the exposed version of the subflow is shown. If make a change to the subflow and want to re-run inspection, you need to expose the new version.</p> <hr/>

3. Select **Run > Inspect Current Flow** or click the **Inspect Current Flow** button on the toolbar.

If you specified one inspection point, the **Inspection Results** pane shows the inspected data in horizontal view. You can change the layout of the view using the toolbar icons above the table. If your inspection data is hierarchical, it cannot be viewed vertically.

AddressLine1	City	City.Type	Country	PostalCode	ProcessedBy	StateProvince	Status
510 S Coit St Flo...	FLORENCE	P	USA	29501	USA	SC	
241 NE C St Wi...	WILLAMINA	P	USA	97396	USA	OR	
2500 Foothill BGr...	GRANTS PASS	P	USA	97526	USA	OR	
3425 N 22nd St D...	BEARSDALE	S	USA	62526	USA	IL	
3425 N 22nd St D...	DECATUR	P	USA	62526	USA	IL	
3205 N 22nd St D...	BEARSDALE	S	USA	62526	USA	IL	
3205 N 22nd St D...	DECATUR	P	USA	62526	USA	IL	
1404 Hertel AveB...	BUFFALO	P	USA	14216	USA	NY	
2005 Sheridan D...	BUFFALO	P	USA	14223	USA	NY	

Note: Date and time data is displayed in the format specified in the type conversion options.

Tip: You can move an inspection point by dragging it to another channel. The inspection data updates automatically.

If you specified two inspection points, the **Inspection Results** pane displays the records as they exist at the two points. The left pane shows the left-most inspection point in the dataflow and the right pane shows the right-most inspection point in the dataflow. Click a record in the right pane to highlight the corresponding record in the left pane to see how the record has changed between the two inspection points.

Point 1				Point			
AddressLine1	PostalCode	StateProvince		AddressLine1	PostalCode	StateProvince	City
510 S Coit St Flo...	29501-5221	SC		510 S Coit St Flo...	29501	SC	FLORENCE
241 NE C St Wi...	97396-2714	OR		241 NE C St Wi...	97396	OR	WILLAMINA
2500 Foothill BGr...	97526-3603	OR		2500 Foothill BGr...	97526	OR	GRANTS PASS
3425 N 22nd St D...	62526-2107	IL		3425 N 22nd St D...	62526	IL	BEARSDALE
3205 N 22nd St D...	62526-2106	IL		3425 N 22nd St D...	62526	IL	DECATUR
1404 Hertel AveB...	14216-2825	NY		3205 N 22nd St D...	62526	IL	BEARSDALE
2005 Sheridan D...	14223-1222	NY		3205 N 22nd St D...	62526	IL	DECATUR
	14223			1404 Hertel AveB...	14216	NY	BUFFALO

Each column represents a field in the dataflow. Columns are arranged in alphabetical order. New fields added between the inspection points are shown in the right pane after the original columns. To reorder columns, click and drag them into the order you want.

These situations influence how the inspection results are displayed for two inspection points:

- If there is a Sorter stage between the two inspection points, the records in the inspection results will be sorted as they were before the Sorter stage. Sorting is ignored in the second inspection point so that you can compare corresponding records from each inspection point side by side.
- If there are stages between the two inspection points that create new records, such as an Aggregator stage, the records shown in the second inspection point will not have a corresponding record in the first inspection point.
- Records that exist at the second inspection point but not at the first are displayed at the bottom of the list of records in the second inspection point.

4. When you update or make changes to the dataflow, click **Run > Inspect Current Flow** to refresh the inspection results.
5. When you close the Inspection Results pane, the inspection data is lost. Similarly, when you close a job, the inspection points and inspection data are lost. To save the inspection results to a file:
 - a) In the inspection results grid, select the rows you wish to save. You can select all data by right-clicking in either pane and clicking **Select All**.
 - b) Select **Copy** from the context menu.
 - c) Open the application into which you want to save the data (for example, Microsoft Excel or Notepad).
 - d) In the application, paste the data.
 - e) Save the file.

Testing a service with Spectrum Management Console


Spectrum Management Console provides a preview feature that allows you to send test data to a service and see the results.

Note: A service must be saved and exposed before it can be tested through 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. Go to the **Services** menu and click the other services containing the service you want to test.
3. Click the service you want to test.
4. Click **Preview**.
5. Enter the input data you want to use for your test. To import data from a file, click the Import button .
6. Click **Run Preview**.

4 - Running a Flow

In this section

Running a Job or Process Flow.....	821
Exposing a Service.....	841
Runtime Options.....	843
Configuring Email Notification for a Flow.....	846



Running a Job or Process Flow

Running a Flow in Spectrum Enterprise Designer

This procedure describes how to manually run a job or process flow.

1. In Spectrum Enterprise Designer, select **File > Open** and open the flow you want to run.
2. Validate a flow prior to running it to ensure that it contains no errors. To validate a flow, select **Run > Validate**.
3. Select **Run > Run current flow**.

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 <i>delimiter</i>	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 <i>property file</i>	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 828.
No	-h <i>host name</i>	Specifies the name or IP address of the Spectrum Technology Platform server.

Required	Argument	Description
No	<code>-i poll interval</code>	Specifies how often to check for completed jobs, in seconds. This applies only in synchronous mode.
Yes	<code>-j job name</code>	A comma-separated list of jobs to run. Job names are case-sensitive. Jobs are started in the order listed.
No	<code>-n email list</code>	Specifies a comma-separated list of additional email addresses for configured job notifications.
No	<code>-o property file</code>	<p>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 843.</p> <p>For example, a flow options properties file for a flow that contains an Assign GeoTAX Info stage may look like this:</p> <pre>OutputCasing=U UseStreetLevelMatching=N TaxKey=T Database.GTX=gsl</pre>
Yes	<code>-p password</code>	The password of the user.
No	<code>-r</code>	<p>Specify this argument to return a detailed report about the job. This option only works if you also specify <code>-w</code>. The report contains this information:</p> <ul style="list-style-type: none"> • 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 <p>For example:</p> <pre>MySimpleJob 4 succeeded 04/09/2019 14:50:47 04/09/2019 14:50:47 100 0 0 </pre> <p>The information is delimited using the delimiter specified in the <code>-d</code> argument.</p>
No	<code>-s port</code>	The socket (port) on which the Spectrum Technology Platform server is running. The default is 8080.

Required	Argument	Description
No	<code>-t timeout</code>	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	<code>-u user name</code>	The login name of the user.
No	<code>-v</code>	Return verbose output.
No	<code>-w</code>	Runs job executor in synchronous mode. This means that job executor remains running until the job completes. If you do not specify <code>-w</code> , 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=Protocol:FileName <code>-S StageName=Protocol:FileName</code>	Overrides the input or output file specified in Read from File or Write to File. For more information, see Overriding Job File Locations on page 823.
No	StageName=Protocol:Schema <code>-S StageName=Protocol:Schema</code>	Overrides the file layout definition specified in Read from File or Write to File with one defined in a schema file. For more information, see Overriding the File Format at the Command Line on page 825.

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 <i>delimiter</i>	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 <i>hostname</i>	Specifies the name or IP address of the Spectrum Technology Platform server.
No	i <i>pollinterval</i>	Specifies how often to check for completed jobs, in seconds. This applies only in synchronous mode.
Yes	j <i>jobname</i>	A comma-separated list of jobs to run. Job names are case-sensitive. Jobs are started in the order listed.
No	n <i>emallist</i>	Specifies a comma-separated list of additional email addresses for configured job notifications.
Yes	p <i>password</i>	The password of the user.
No	r	Returns a delimited list with this information about the job written to standard output: <ul style="list-style-type: none"> • 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)

Required	Argument	Description
		<ul style="list-style-type: none"> • Position 6 - Number of successful records • Position 7 - Number of failed records • Position 8 - Number of malformed records • Position 9 - Currently unused <p>The information is delimited using the delimiter specified in the <code>-d</code> argument. For example:</p> <pre>MySimpleJob 4 succeeded 04/09/2019 14:50:47 04/09/2019 14:50:47 100 0 0 </pre>
No	<code>s port</code>	The socket (port) on which the Spectrum Technology Platform server is running. The default is 8080.
No	<code>t timeout</code>	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	<code>u username</code>	The login name of the user.
No	<code>v</code>	Return verbose output.
No	<code>w</code>	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.

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 <i>DelimiterCharacter</i>	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	-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 <i>PropertyFile</i>	Specifies a path to a property file. For more information on property files, see Using a Process Flow Property File on page 831.
No	-h <i>HostName</i>	Specifies the name or IP address of the Spectrum Technology Platform server.
No	-i <i>PollInterval</i>	Specifies how often to check for completed jobs, in seconds. The default is "5".
Yes	-p <i>Password</i>	The password of the user. Required.
Yes	-r <i>ProcessFlowNames</i>	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 <i>Port</i>	The socket (port) on which the Spectrum Technology Platform server is running. The default is 8080.
No	-t <i>Timeout</i>	This option is deprecated and will be ignored.
Yes	-u <i>UserName</i>	The login name of the user. Required.
No	-v <i>Verbose</i>	Return verbose output where <i>Verbose</i> 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.

Required	Argument	Description
No	-w <i>WaitToComplete</i>	This option is deprecated and will be ignored.
No	<i>StageName=FileName</i>	Overrides the input or output file specified in the job. For more information, see Overriding Process Flow File Locations .

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 829 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 /dcb/flow.properties -r MyFlow2
```

In the above example, the process flow `MyFlow2` would take precedence over a process flow specified in the properties file.

Scheduling a Flow

A flow schedule runs a job or process flow automatically at a specific time. You can schedule a flow to run once, or set up recurring execution.

1. If you have not already done so, expose the job or process flow.

You can expose jobs and process flows by opening the job or process flow in Spectrum Enterprise Designer and selecting **File > Expose/Unexpose and Save**.

2. Open Spectrum Management Console.
3. Go to **Execution** then click **Scheduling**.
4. Click **Add** to create a new schedule or, if you want to modify an existing schedule, choose the schedule and click **Modify**.
5. In the **Add Task** or **Modify Task** window, choose the settings for this task.
 - **Task Name** - The name you want to give to this scheduled task. This is the name that will be displayed in the task listing.
 - **Flow type** - Choose the type of process you are scheduling, either a job or a process flow.
 - **Flow name** - Select the job or process flow that you want to schedule. Only jobs and process flows that are saved and exposed are available here. If the job or process flow that you want is not shown, open the job or process flow in Spectrum Enterprise Designer then select **File > Expose/Unexpose and Save**.
 - **Enable task** - Check this box to run the job or process flow at the specified time. Clear this box to suspend the schedule.
 - **Schedule** - Specify the date and time you want the job or process flow to run.
6. If the flow uses files for input or output, those files 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 do one of these steps:

Option	Description
Option 1: Modify the flow	<p>Move the file to the Spectrum Technology Platform server or file server then modify the flow:</p> <ol style="list-style-type: none"> Open the flow in Spectrum Enterprise Designer. Double-click the source or sink stage. In the File name field, click the browse button. Click Remote Machine then select the file you want. <p>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.</p>
Option 2: Override the flow file location when this schedule runs	<p>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).</p> <ol style="list-style-type: none"> Click Options. Under Stage file locations select the stage that references a local file. Click Modify and select the file on the Spectrum Technology Platform server.

- If you want the job or process flow to run on a recurring schedule, check the **Task recurrence** check box then click the **Recurrence** button and complete the fields.
- If the flow has been configured for email notification, you can specify additional recipients for the notifications that will be sent when the flow runs.
 - Click **Options**.
 - Under Notification, click **Add**.
 - Enter the email address to receive notification. For example, `me@mycompany.com`.
 - Click **OK**.

Note: Notification must be configured in Spectrum Management Console in order for email notifications to work. In addition, verify that the flow has been configured to support notification. To do this, open the flow in Spectrum Enterprise Designer, select **Edit > Notifications**.

- Click **OK**.


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.

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.

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 terminate. 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:

Terminate jobs containing this many malformed records	Select this option to terminate jobs if the input data contains one or more malformed records. Enter the number of
--	--

Do not terminate flows with malformed records

malformed records that you want to trigger the termination of the job. The default is 1.

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 within 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.

Exposing a Service

Exposing a Service as a Web Service

Spectrum Technology Platform services can be made available as RESTful and SOAP web services. To make a service available on your server as a web service:

1. Open Spectrum Enterprise Designer.
2. Open the service that you want to expose as a web service.
3. Go to **Edit > Web Service Options**.
4. To make the service available as a SOAP web service, check the box **Expose as SOAP web service**.
5. To make the service available as a REST web service, check the box **Expose as REST web service** and complete these steps.
 - a) If you want to override the default endpoint, specify the endpoint you want to use in the **Path** field.

Specifying a path is optional. By default, a REST web service's endpoint is:

```
http://server:port/rest/service_name/results.qualifier
```

If you want to use a different endpoint, the path you specify is added after the service name. For example, if you specify `Americas/Shipping` in the **Path** field, your JSON endpoint would be something like this:

```
http://myserver:8080/rest/MyService/Americas/Shipping/results.json
```

You can use fields and options from the flow as variable names in the path by clicking the **Insert variable** drop-down menu and selecting the field or option you want to use. The

variable is represented in the path using the notation `${Option.Name}` for flow options and `${Data.Name}` for flow fields.

- b) By default REST web services support the GET method and return data in XML and JSON formats. You can define additional HTTP methods and output formats by clicking **Add** to add a resource to the web service.

When you add a resource, you can choose the HTTP method (**GET** or **POST**). The supported data formats are listed below. You may not have all these formats available to you because some formats are only available if you have certain modules installed on your Spectrum Technology Platform server.

XML The default XML format. Use this format if you want to use XML as the format for requests and responses, and there is no specialized XML format for the kind of data you want to process.

JSON The default JSON format. Use this format if you want to use JSON as the format for requests and responses, and there is no specialized JSON format for the kind of data you want to process.

GeoJSON A specialized JSON format that is appropriate for services that handle geographic data. Support is provided only for Geometry and for these native platform types:

- boolean
- double
- float
- integer
- bigdecimal
- long
- string
- date
- time
- datetime
- timespan

If you try to expose a flow with any other type, you will not be able to specify GeoJSON (an error will appear at design-time). Also, GeoJSON only allows a single geometry. If the output contains multiple geometry fields, the system will search for a field called "geometry" followed by a field called "obj." If those fields do not exist, the first geometry field will be selected.

- c) Click **OK**.

The new resource is added to the web service.

6. Click **OK** when you are done configuring the web service options.
7. Click the gray light bulb in the tool bar to expose the service.

When a flow is exposed the light bulb button in the Spectrum Enterprise Designer tool bar indicates that the flow is exposed as shown here:



To verify that the service is now exposed as a web service, go to one of the following URLs:

- For REST: `http://server:port/rest`
- For SOAP: `http://server:port/soap`

Where *server* is the name or IP address of your Spectrum Technology Platform server and *port* is the port used for HTTP communication.

Exposing a Service for API Access

In order for a service to be accessible from through the Spectrum Technology Platform API, you must expose the service.

1. Open the service in Enterprise Designer.
2. Click the gray light bulb in the tool bar to expose the service.

When a flow is exposed the light bulb button in the Spectrum Enterprise Designer tool bar indicates that the flow is exposed as shown here:



Runtime Options

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 845.

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.

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**.

Deleting flow Runtime Options

Flow runtime options enable you to set stage options at runtime. The stage options can be set when calling the job through a process flow or through the job executor command-line tool. This procedure describes how to remove a flow runtime option so that the option can no longer be set at runtime.

1. Open the job, service, or subflow.
2. Click the **Dataflow Options** icon or click **Edit > Dataflow Options**. The **Dataflow Options** dialog box appears.
3. Highlight the option you want to delete and click **Remove**.

Configuring Email Notification for a Flow

A flow can be configured to send an email notification containing job status information. For example you may want to send an email alert if a flow fails. The email notification can contain information such as the flow name, the start and end time, the number of records processed, and more.

Note: A mail server must be configured in Spectrum Management Console before you can set up notification for a flow. For more information, see the *Spectrum Technology Platform Administration Guide*.

1. With a flow or process flow open in Spectrum Enterprise Designer, select **Edit > Notifications**.
2. Click **Add**.
3. In the **Send Notification To** field, enter the email address to which notifications should be sent.
4. Select the events you want to be notified about.
5. In the **Subject** field, enter the text you would like to appear in the subject line of the email.
6. In the **Message** field, enter the text you would like to appear in the body of the email.

You can choose to include information about the job in the email by clicking **Click Here to Insert a Field in the Subject or Message**. Some examples of job information are: start time, end time, and number of records failed.

7. Click **Preview** if you wish to see what the notification will look like.
8. Click **OK**. The **Notifications** dialog box will reappear with the new notification listed.
9. Click **OK**.

5 - Combining Flows into a Process Flow

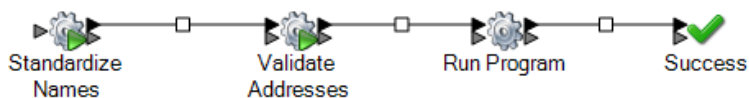
In this section

Introduction to Process Flows.....	849
Designing Process Flows.....	849



Introduction to Process Flows

A process flow runs a series of activities such as jobs and external applications. Each activity in the process flow runs after the previous activity finishes. Process flows are useful if you want to run multiple flows in sequence or if you want to run an external program. For example, a process flow could run a job to standardize names, validate addresses, then invoke an external application to sort the records into the proper sequence to claim postal discounts. Such a process flow would look like this:



In this example, the jobs Standardize Names and Validate Addresses are exposed jobs on the Spectrum Technology Platform server. Run Program invokes an external application, and the Success activity indicates the end of the process flow.

Designing Process Flows

Creating a Process Flow

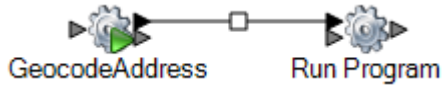
To create a process flow use Spectrum Enterprise Designer to create a sequence of activities that run jobs and or external applications.

1. Open Spectrum Enterprise Designer.
2. Select **File > New > Process Flow**.
3. Add the first action you want the process flow to perform. You can chose one of these options:
 - To execute a job, drag the job's icon from the **Activities** folder in the palette to the canvas.
 - To execute an external program, drag a Run Program icon from the **Activities** folder in the palette to the canvas.
4. Add the second action you want the process flow to perform.

You can add a job by dragging a job's icon to the canvas, or add an external program by dragging a Run Program icon to the canvas.

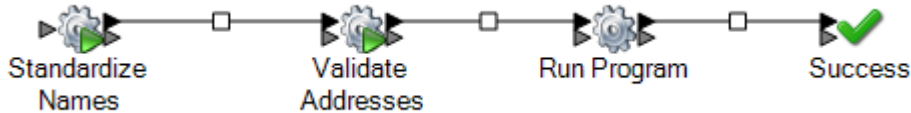
5. Connect the two activities by clicking the gray triangle on the right side of the first icon and dragging it to connect to the gray icon on the left side of the second icon.

For example, if you have a process flow that first runs a job named GeocodeAddress then runs an external program, your process flow would look like this:



6. Add additional activities as needed.
7. When you have added all the activities you want to run in the process flow, drag a Success activity to the canvas and connect it to the last activity in the process flow.

For example, this process flow contains two jobs ("Standardize Names" and "Validate Addresses") and one Run Program activity. At the end of this process flow is the Success activity:



8. Double-click the activities you placed on the canvas to configure their runtime options. You can also double-click the connection between activities to configure transition options.

Using a Variable to Reference a File

In a process flow, variables are useful if you want multiple activities in the process flow to reference the same file. Using a variable, can define the file in one place, then reference the variable in all downstream activities that need to reference the file. If the file ever changes, you can modify the variable definition without having to modify all the downstream activities.

When you add a job activity to a process flow, the activity automatically creates variables for each source and sink in the dataflow. If there are files you want to use in the process flow that are not defined in the source or sink of a job, you can create variables.

When you add a Run Program activity, no variables are created by default. If you want to use variables with a Run Program activity you must create them.

This procedure describes how to create a variable in a job activity or a Run Program activity.

1. Open the process flow in Spectrum Enterprise Designer.
2. Double-click the job activity or Run Program activity where you want to define the variable.

Note: Variables can only be referenced by activities that follow the activity where you define the variable, so be sure to define the variable in an activity that precedes the activities where you want to use the variable.

3. Click the **Variables** tab.
4. Create the variable.

Option	Description
<p>To create a new variable for an input file...</p>	<p>Next to Inputs click Add. In the Name field enter a name for the variable. This is the name that downstream activities will reference. In the Location field choose one of these options:</p> <p>Use file specified in job Choose this option to use the file defined in the source stage in the job. This option is only available if you are defining a variable for a job activity.</p> <p>Browse for file on the server Choose this option if you want to select a file to assign to this variable.</p> <p>Reference an upstream activity's file Choose this option if you want to use a file assigned to an existing variable from an upstream stage.</p>
<p>To create a new variable for an output file...</p>	<p>Next to Inputs click Add. In the Name field enter a name for the variable. This is the name that downstream activities will reference. In the Location field choose one of these options:</p> <p>Browse for file on the server Choose this option if you want to select a file to assign to this variable.</p> <p>Temporary file managed by the server Choose this option if you want this variable to reference a temporary file that will be automatically created and deleted as needed. This option is useful in cases where a file used only as an intermediate step in a process flow and is not needed once the process flow completes.</p>

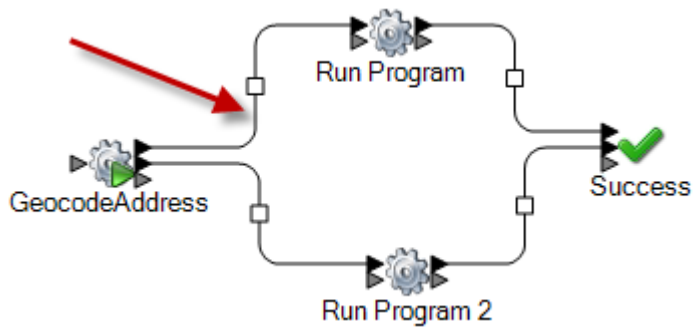
5. Click **OK** to close the **Add Variable** window.
6. Click **OK** to close the activity options window.
7. To reference the variable in a downstream activity:
 - a) Double-click the activity that you want to reference the variable.
 - b) Select the input stage that you want to have reference the variable and click **Modify**.
 - c) In the **Location** field, select Reference an upstream activity's file....

Adding Conditional Logic to a Process Flow

You can add conditional logic to a process flow so that different activities run based on the return code of a preceding activity. For example, you could execute one activity if a job returns a return code of 1 and another activity if a job returns a return code of 0. In this way you can build conditional branching into your process flow.

1. Open the process flow in Spectrum Enterprise Designer.
2. Double-click the transition between two activities of the flow.

A *transition* is the line that connects two activities. For example, the line between the GeocodeAddress activity and the Run Program activity shown here is a transition:



The **Transition Options** window appears.

3. Select the type of transition you wish to add.

- | | |
|--------------------|---|
| Simple | Select this option if you always want this path in the process flow to run. |
| Conditional | Select this option if you only want this path in the process flow to run if the upstream activity returns a specific return code or return codes, or a range of return codes. |
| Otherwise | Select this option if you want this path in the process flow to run only if the conditions in the other transitions leading from the activity are not met. |

Note: Only one **Otherwise** transition can exist among the transitions leading from an activity.

4. Click **OK**.
5. To configure which transitions trigger an activity, right-click the activity, select **Input modes**, then choose one of the following:

- | | |
|--------------------|---|
| First Input | The first transition coming into this activity, whether through a Simple , Conditional , or Otherwise transition, triggers the execution of the activity. Other transitions are ignored. |
|--------------------|---|

All Inputs The activity does not run unless all transitions coming into this activity are taken.

- To configure which transitions leading out of an activity are taken, right-click the activity, select **Output modes**, then choose one of these options:

First Output The first transition that evaluates to true is taken. Other transitions are ignored, even if their conditions evaluate to true.

All Outputs All transitions that evaluate to true are taken.

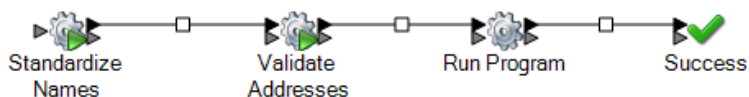
Deleting a Process Flow

- Go to **File > Manage**. The **Manage** dialog box will appear.
- Right-click on the process flow you want to delete and select **Delete**.
- Click **OK**.

Activities

Job

A job activity runs a job as part of a process flow. This example shows a process flow that runs two job activities: Standardize Names and Validate Addresses.



To add a job activity to a process flow, drag the job activity from the Activities folder in the palette to the canvas.

Note: In order for a job to be available to use in a process flow, the job must be exposed. If the job you want is not exposed, open the job in Spectrum Enterprise Designer and select **File > Expose**.

Double-click the job activity to configure the **Options** tab and the **Variables** tab.

Options Tab

The Options tab displays the runtime options available for the job. You can change the runtime options so that when the job runs the options you specify here are used for the job. For example, if

one of the job's dataflow options is controls the units to use for distance and defaults to miles, you could override that option here and have distance returned in kilometers instead when the job is executed through this process flow.

Variables Tab

The Variables tab displays the source and sink stages in the dataflow. You can override the input and output files specified in the dataflow so that when the job runs in this process flow, different input or output files are used.

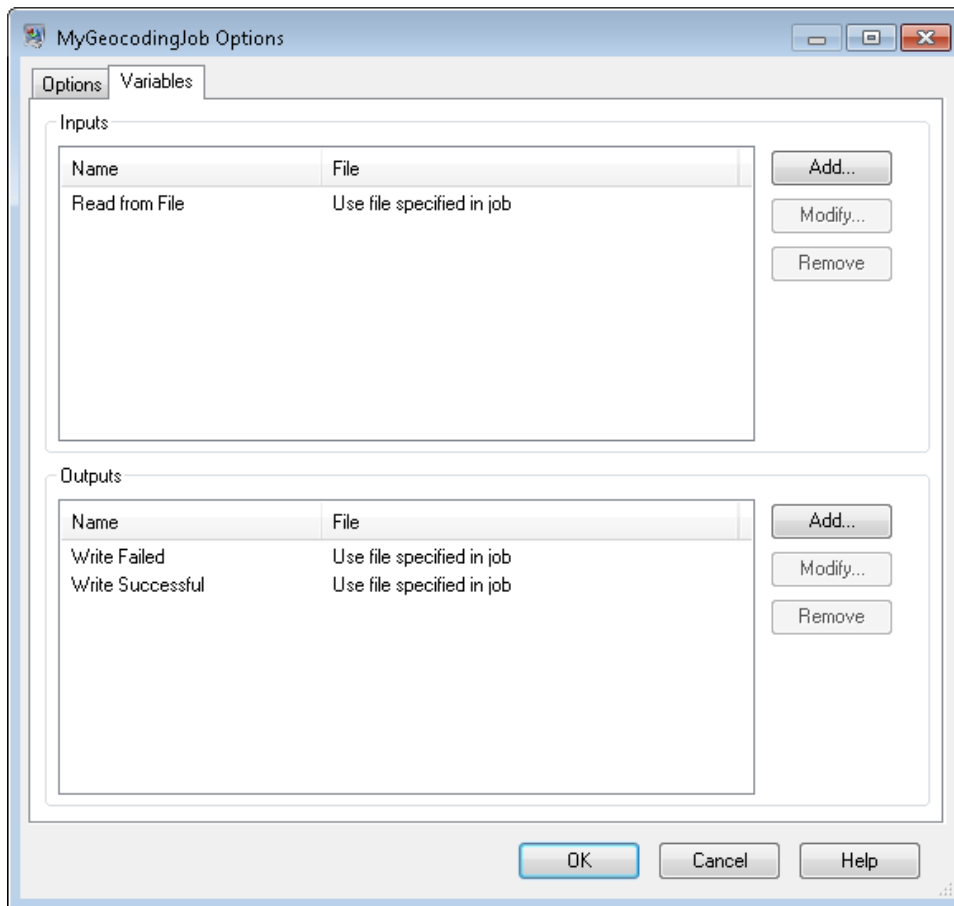
Overriding Input and Output Files

By default, a job executed in a process flow uses the input and output files defined in the job's source and sink stages. You can, however, override the input and output files defined in the job so that when the process flow runs, the job will use the input or output file you specify in the job activity in the process flow instead of the file specified in the job's source or sink stages. You can override input and output files by specifying a specific file or by using a variable to refer to a file defined in an upstream activity.

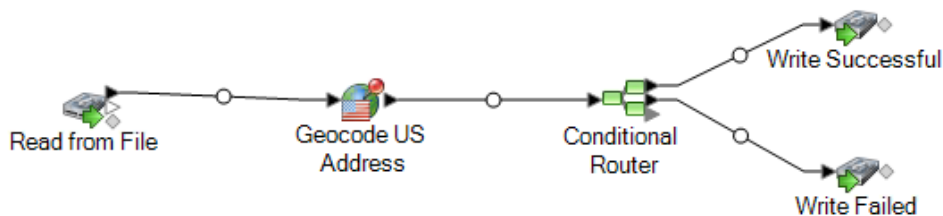
1. Open the process flow in Spectrum Enterprise Designer.
2. Double-click the job activity for which you want to override an input or output file.
3. Click the **Variables** tab.

On the **Variables** tab, the variables listed under **Inputs** correspond to the source stages in the job. The variables listed under **Outputs** correspond to the sink stages in the job.

For example, say you have a process flow that contains a job activity for a job named MyGeocodingJob. On the **Variables** tab of the activity options you see this:



Each variable listed corresponds to the name of a source or sink stage in the MyGeocodingJob dataflow. In this example the **Variables** tab shows one source (Read from File) and two sinks (Write Failed and Write Successful). If you were to open up the MyGeocodingJob dataflow in Spectrum Enterprise Designer, you would see something like this:



4. On the **Variables** tab, select the source stage or output stage that you want to override and click **Modify**.
5. the **Location** field, choose one of the following to override an input or output file.
To override an input file:

Choose this	To
Use file specified in job	Use the file defined in the source stage in the job.
Browse for file on the server	Override the file defined in the job and use a different file that you choose.
Reference an upstream activity's file	Override the file defined in the job with a file whose name and location is defined in an upstream activity's Read from File or Write to File stage or an upstream activity's variable. Use this option if the output file from a previous activity is the input for this activity. The advantage of this option is that if the upstream activity's Write to File stage is ever modified to point to another file, this activity will still point to the correct file. Another advantage is that you do not need to know the file path and name of upstream activities' input file to point to it.

To override an output file:

Choose this	To
Browse for file on the server	Override the file defined in the job and use a different file that you choose.
Temporary file managed by the server	Make this variable reference a temporary file that will be automatically created and deleted as needed. This option is useful in cases where a file used only as an intermediate step in a process flow and is not needed once the process flow completes.

- Click **OK** to close the **Modify Variable** window.
- Click **OK** to close the activity options window.

When the process flow runs, the job will use the files you specified in the process flow activity instead of the file specified in the job itself.

Clear Cache

A Clear Cache activity clears the global cache data as a part of process flow. It does not delete the cache but only clears the cache data.

- Drag the **Clear Cache** activity to the canvas.
- Double-click the Clear Cache.
- Select the cache. You can also select multiple caches to clear their data.

4. Run the process flow.

Execute SQL

An Execute SQL activity allows you to run the SQL statements both before and after you run a dataflow or an external program.

Load to Hive

Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis. To use Hive to query the underlying data source, use its own query language, HiveQL.

Hive supports these Hadoop file formats:

- TEXTFILE
- SEQUENCE FILE
- ORC
- PARQUET
- AVRO

Note: The AVRO file format is supported in Hive version 0.14 and higher.

The **Load to Hive** activity allows you to load data into a Hive table using a JDBC connection. Using this connection, data is read from a specified Hadoop file and loaded to either an existing table of a selected connection, or to a newly created table in the selected connection.

To load the data to a new table, the schema of the table needs to be defined. Spectrum does not support hierarchical data, even though Hive supports it.

Note: The stage supports reading data from and writing data to HDFS 3.x and Hive 2.1.1. The support includes:

- Connectivity to HDFS and Hive from Spectrum on Windows
- Support and connectivity to Hadoop 3.x from Spectrum with high availability
- Kerberos-enabled HDFS connectivity through Windows
- Support and connectivity to Hive version 2.1.1 from Spectrum with high availability
- Support of Datetime datatype in the Parquet file format
- Support to Read and Write from Hive DB (JDBC) via Model Store connection

Also see [Configuring HDFS Connection for HA Cluster](#) and [Best Practices for connecting to HDFS 3.x and Hive 2.1.1](#).

Creating a Hive Connection

1. Open the **Load to Hive** activity.

2. From the **File name** field, enter the name of the file, which is to be read. Click Browse [...] to select the file to be read.
3. From the **File type** field, select the format of the file to be read. The default file format selected is `Delimited`.

If the **File type** selected is either `Delimited` or `Sequence`, the fields **Field Separator** and **Record Separator** are displayed. Else, they are not displayed.

4. In the **Field Separator** field, select the character that separates each consecutive field of a record.
5. Select the connection for the Hive database you want to use in the **Connection** field.
 - a) To add, modify, and delete connections, click **Manage**.
The **Database Connection Manager** window opens.
 - b) Click **Add** to create a new connection, or **Modify** to edit an existing connection.
The **Connection Properties** window opens.
 - c) Enter the **Connection Name**.
 - d) In the **Database Driver** field, select a Hive database driver for the connection.
 - e) Specify all the details of the connection, namely **user**, **password**, **host**, **port**, and **instance**.
 - f) To test the connection details, click **Test**.
 - g) If the connection test is successful, click **OK**.
The **Connection Properties** window closes.
 - h) Click **OK**.
The **Database Connection Manager** window closes.
6. In the **Table/View** field, select the table you wish to write to, or type in the name of a new table to be created.

If you create a new table in the **Table/View** field, the **External** check box gets enabled. Else, if you select an existing table, the **External** check box remains disabled.

7. To create the new table external to the Hive database, check the **External** check box.

Important: In case of External tables: You cannot overwrite records and you cannot add new records. You are allowed to create new external tables and populate those with records. If you select a file placed in a particular folder, all files placed in that folder are automatically selected. Ensure that all files placed in the particular folder have the same format. Learn more about Hive EXTERNAL tables [here](#).

8. To overwrite all existing records of the table, check the **Overwrite** check box. This deletes existing records of the selected table, and adds the records read from the file to the table.
9. The grid displays the names and data types of the columns of the selected table.

If you have specified a new table in the **Table/View** field, use the **Add**, **Modify** and **Remove** buttons beside the grid to add columns to define the table, and specify their respective data types. Use the **Move Up** and **Move Down** buttons to specify the sequence of the table columns.

Note: The **Add**, **Modify**, **Remove**, **Move Up**, and **Move Down** buttons remain disabled if you select an existing table in the **Table/View** field.

Important:

- a. Ensure the data types of all the fields in the file match the data types of the respective table columns, unless all data types are of String type. Else, the data load may result in inconsistent data.
- b. Ensure the number of fields in the file match the number of table columns. Else, the data in the extra fields in the file are discarded.
- c. Hive accepts names of tables and columns in small case only. If you enter the names using block letters, Hive converts them to small case. The resultant schema displays all names in small case.

10. Click **OK**.

Note: If you opt to create a new table and define its columns, the same is created at runtime. The **Load to Hive** activity is only to design the table structure. At runtime, the designed table is created and the data read from the file is written into it.

Run Program

The Run Program activity runs an external application as part of a process flow.

Table 151: Run Program Options

Option Name	Description
Program name	The path to the executable you wish to run.
Arguments	Specifies command line arguments to pass to the program specified in the Program name field. Separate multiple arguments with spaces. You can use variables defined on the Variables tab as arguments by clicking Insert Variable . For more information about variables, see Using a Variable to Reference a File on page 850.
Time out (in seconds)	Specifies an amount of time to wait for the program specified in the Program name field to respond. If the program is unresponsive for the amount of time specified the process flow will fail.

Option Name	Description
Environment variables	<p>Specifies environment variable values to use when running this program. If you specify values here the program will use these environment variables instead of those specified on your system. Otherwise, it will use the environment variables specified on your system. Note that if the program you are calling uses multiple environment variables you must either define values for all of them or none of them. Specifying values here does not change the environment variable definitions on your system.</p> <p>Click Add and enter the name of the variable in the Variable Name field; for example: "JAVA_HOME". Enter the value of the variable in the Variable Value field; for example: C:\Program Files\Java\jdkversion. Instead of entering a value you can click Insert Variable to set it to the value of a variable defined in on the Variables tab.</p>

Specifying Input and Output Files

You can call an external application from a process flow by using a Run Program activity. You can specify the file that contains the data you want to send to the external application as well as the output file you want the external application to write to.

1. In a process flow, double-click a Run Program activity.
2. Click the **Variables** tab.
3. To specify an input file,
 - a) Click the **Add** button under the **Inputs** section.
 - b) In the **Name** field, enter a meaningful name for this file. The name can be anything you choose.
 - c) In the **Location** field choose one of the following:

Browse for file on the server	Choose this option to go to the input file you want and select it.
Reference an upstream activity's file	Choose this option if you want to use a file assigned to an existing variable from an upstream stage.
 - d) Click **OK**.
4. To specify an output file,
 - a) Click the **Add** button under the **Outputs** section.
 - b) In the **Name** field, enter a meaningful name for this file. The name can be anything you choose.
 - c) In the **Location** field choose one of the following:

Browse for file on the server	Choose this option if you want to go to an output file and select it.
--------------------------------------	---

Temporary file managed by the server Choose this option if you want the output from the program to be written to a temporary file that will be automatically created and deleted as needed. This option is useful in cases where a file used only as an intermediate step in a process flow and is not needed once the process flow completes.

- d) Click **OK**.
- 5. Click **OK** to close the **Run Program Options** window.

Using a Control File with an External Program

You can call an external application from a process flow by using a Run Program activity. In doing so you can use a control file that contains configuration settings for the external application. For example, you could call VeriMove™ with a Run Program activity and specify a control file to use during execution.

1. In a process flow, double-click a Run Program activity.
2. Click the **Variables** tab.
3. In the **Control Files** section, click **Add**.
4. In the **Name** field, give the control file a name. The name can be anything you choose.
5. In the **Contents** field, specify the contents of the control file.

To reference a file using a variable defined under **Inputs** or **Outputs**, click **Insert Variable**.

6. Click **OK** to close the **Add Control File** window.
7. Click the **Options** tab.
8. Click **Insert Variable**.
9. Select the name of the control file you created then click **OK**.

A variable that points to your control file is added to the **Arguments** field.

10. If necessary modify the **Arguments** field to use the necessary command line arguments to indicate a control file. Your external application documentation has more information.
11. Click **OK** to close the **Run Program Options** window.

Success

A Success activity indicates the end of a process flow. A process flow must have at least one Success activity.

6 - Creating Reusable Flow Components

In this section

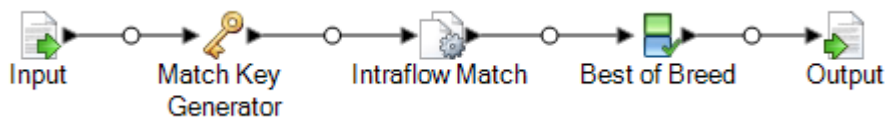
Introduction to Subflows.....	863
Using a Subflow as a Source.....	863
Using a Subflow in the Middle of a Flow.....	864
Using a Subflow as a Sink.....	865
Modifying a Subflow.....	866
Deleting a Subflow.....	867
Exposing and Unexposing a Subflow.....	867
Converting a Stage to a Subflow.....	867



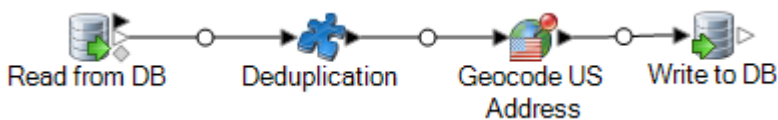
Introduction to Subflows

Subflows are snippets of flows.

A subflow is a dataflow that can be reused within other dataflows. Subflows are useful when you want to create a reusable process that can be easily incorporated into dataflows. For example, you might want to create a subflow that performs deduplication using certain settings in each stage so that you can use the same deduplication process in multiple dataflows. To do this you could create a subflow like this:



You could then use this subflow in a dataflow. For example, you could use the deduplication subflow within a dataflow that performs geocoding so that the data is deduplicated before the geocoding operation:



In this example, data would be read in from a database then passed to the deduplication subflow, where it would be processed through Match Key Generator, then Intraflow Match, then Best of Breed, and finally sent out of the subflow and on to the next stage in the parent dataflow, in this case Geocode US Address. Subflows are represented as a puzzle piece icon in the dataflow, as shown above.

Subflows that are saved and exposed are displayed in the **User Defined Stages** folder.

Using a Subflow as a Source

You can use a subflow as the first stage in a flow to read data from a source and even perform some processing on the data before passing it to the parent flow. You can create a subflow that is as simple as a single source stage that is configured in a way that you want to reuse in multiple flows, or you could create a more complex subflow that reads data then processes it in some way before passing it to the parent flow.

1. In Spectrum Enterprise Designer, click **File > New > Dataflow > Subflow**.
2. Drag the appropriate data source from the palette to the canvas and configure it.

For example, if you want the subflow to read data from a comma-separated file, you would drag a Read from File stage to the canvas.

3. If you want the subflow to process the data in some way before sending it to the parent flow, add additional stages as needed to perform the preprocessing you want.
4. At the end of the flow, add an Output stage and configure it.

This allows the data from the subflow to be sent to the parent flow.

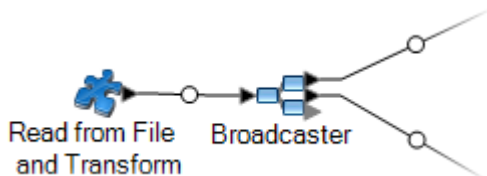
For example, if you created a subflow that reads data from a file then uses a Transformer stage to trim white space and standardize the casing of a field, you would have a subflow that looks like this:



5. Double-click the Output stage and select the fields you want to pass into the parent flow.
6. Select **File > Save** and save the subflow.
7. Select **File > Expose** to make the subflow available to include in flows.
8. In the flow where you want to include the subflow, drag the subflow from the palette to the canvas.
9. Connect the subflow to the flow stage you want.

Note: Since the subflow contains a source stage rather than an Input stage, the subflow icon only has an output port. It can only be used as a source in the flow.

The parent flow now uses the subflow you created as input. For example, if you created a subflow named "Read from File and Transform" and you add the subflow and connect it to a Broadcaster stage, your flow would look like this:



Using a Subflow in the Middle of a Flow

You can use a subflow in the middle of a flow to perform processing that you want to make reusable in other flows. In effect, the subflow becomes a custom stage in your flow.

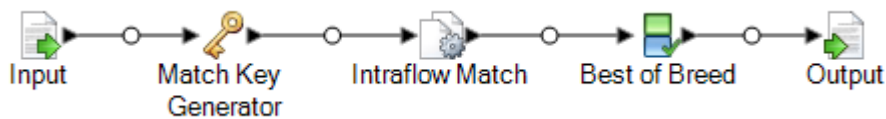
1. In Spectrum Enterprise Designer, click **File > New > Dataflow > Subflow**.
2. Drag an Input stage from the palette to the canvas.

This allows data from the parent flow to be sent into the subflow.

3. Double-click the Input stage and add the fields that the subflow will receive from the flow in which it is used.
4. After configuring the Input stage, add additional stages as needed to perform the processing that you want.
5. At the end of the flow, add an Output stage.

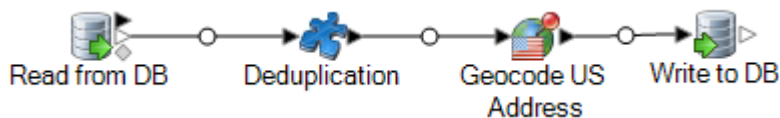
This allows the data from the subflow to be sent to the parent flow.

For example, you might want to create a subflow that performs deduplication using certain settings in each stage so that you can use the same deduplication process in multiple flows. To do this you could create a subflow like this:



6. Select **File > Save** and save the subflow.
7. Select **File > Expose** to make the subflow available to include in flows.
8. In the flow where you want to include the subflow, drag the subflow from the palette to the canvas.
9. Connect the subflow to the flow stage you want.

For example, you could use the deduplication subflow within a flow that performs geocoding so that the data is deduplicated before the geocoding operation:



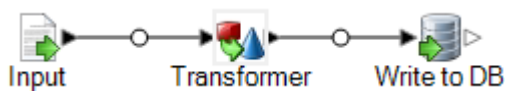
Using a Subflow as a Sink

You can use a subflow as the last stage in a flow to write data to a file or database and even perform some processing on the data before writing the data to the output destination. You can create a subflow as simple as a single sink stage that is configured in a way that you want to reuse in multiple flows, or you could create a more complex subflow that processes data in some way before writing it to the output destination.

1. In Spectrum Enterprise Designer, click **File > New > Dataflow > Subflow**.
2. Drag an Input stage from the palette to the canvas.
3. Double-click the Input stage and add the fields that the subflow will receive from the flow in which it is used.

4. After configuring the Input stage, add additional stages as needed to perform the post-processing that you want.
5. At the end of the subflow, add the appropriate sink.

For example, if you created a subflow that uses a Transformer stage to trim white space and standardize the casing of a field then writes it to a database, you would have a subflow that looks like this:



6. Select **File > Save** and save the subflow.
7. Select **File > Expose** to make the subflow available to include in flows.
8. In the flow where you want to include the subflow, drag the subflow from the palette to the canvas and connect it to the last stage in the flow.

Note: Since the subflow contains a sink stage rather than an Output stage, the subflow icon only has an input port. It can only be used as a sink in the flow.

The parent flow now uses the subflow you created as a sink. For example, if you created a subflow named "Transform and Write to DB" and you add the subflow and connect it to a Geocode US Address stage, your flow would look like this:



Modifying a Subflow

1. Open the subflow on the canvas.
2. Before modifying the subflow, you may want to consider how the change will impact the flows using the subflow. To see which flows are using the subflow, select **Tools > Used By**.
3. Modify the subflow as needed.
 - When you delete an Input or Output stage or add an additional Input or Output stage, Spectrum Enterprise Designer displays a warning message reminding you that other flows are using the subflow and giving you the option of seeing which flows use the subflow. If you continue saving the reusable stage, Spectrum Enterprise Designer will conceal all flows used by the subflow.
 - If you change a subflow in another way, such as by changing a file name or the stage configurations, Spectrum Enterprise Designer will display a warning message reminding you that other flows are using the subflow and give you the option of seeing which flows use the subflow. You can continue without concealing those flows.

4. When you are done making your changes, select **File > Save**.
5. Select **View > Refresh** in order for the changes to be reflected in the parent flow.

Note: If you have more than one version of the subflow, remember that the version that is used in the parent dataflow is the exposed version. When you make a change to a subflow, be sure to expose the most recent version in order for your changes to take effect in the dataflows that use the subflow.

Deleting a Subflow

If you try to delete an exposed subflow, Spectrum Enterprise Designer displays a warning message reminding you that other flows are using the subflow you are about to delete. If you continue to delete the subflow, Spectrum Enterprise Designer conceals all connected flows.

Exposing and Unexposing a Subflow

In order for a subflow to be available for use within a dataflow the subflow must be exposed. To expose a subflow, open the subflow in Spectrum Enterprise Designer and go to **File > Expose/Unexpose and Save**. This will make the subflow available for use in other dataflows.

Note: If you have more than one version of the subflow, remember that the version that is used in the parent dataflow is the exposed version. When you make a change to a subflow, be sure to expose the most recent version in order for your changes to take effect in the dataflows that use the subflow.

To unexpose a subflow, open the subflow in Spectrum Enterprise Designer and select **File > Expose/Unexpose and Save**. When you unexpose a subflow, Spectrum Enterprise Designer displays a warning message reminding you that other dataflows are using the subflow you are about to alter. If you continue to unexpose the subflow, Spectrum Enterprise Designer unexposes all dataflows that use the subflow.

Converting a Stage to a Subflow

1. Create a new job, service, or subflow.

2. Add the stage you would like to include in the job, service, or subflow.
3. If you wish to configure the stage at this point, right-click the stage and select **Options**. Then configure the stage options as desired and click **OK**.
4. Right-click the stage you want to convert and select **Convert Stage to Subflow**. The **Save As** dialog box appears.
5. Enter the name you want to give the subflow and click **OK**, then save the service. The name must be unique to the system. Three things happen:
 - The system creates a new subflow that includes:
 - the stage you selected
 - a flow input for each input port on the stage
 - a flow output for each output port on the stage
 - connections between the stage and its inputs and outputs
 - The system replaces your selected stage with the new subflow.
 - The system exposes the new subflow. You will see it in the Server Explorer and in the User Defined Stages section of the toolbox.

After you have created a subflow and used it in other flows, you can see what other flows are using the subflow. Open the subflow and go to **Tools > Used By**. (Alternately, you can right-click the subflow in Server Explorer and select **Used By**.) This will show a list of flows that use the current subflow, allowing you to see which flows would be affected if you changed the current subflow.

7 - Sample Flows

In this section

Introduction.....	870
Integration between SugarCRM OnPremises and Microsoft Dynamics 365 Online.....	872
Integration between Salesforce and Oracle Eloqua.....	874



Introduction

This section describes the end-to-end process of data migration using sample flows.

Note: These sample templates are part of the Spectrum installer file and do not work standalone. There is no further support available for these samples.

Locating the Sample Template

We provide a set of preconfigured templates that demonstrate Spectrum migration capabilities. This provides a starting point to create flows. Create two or more connections and import these flows into your system to demonstrate the migration capabilities. The sample template is shipped with the Spectrum Technology Platform installer zip.

- Depending on your installation directory, you can find the sample template at this path:

```
Program  
Files\Precisely\Spectrum\server\modules\discovery\connectors\samples
```

This folder contains all the required files to configure the data flow.

Creating Connections

You need to create two connections to import and deploy required files to configure the data flow. Follow these steps to create connections:

1. Install the latest Spectrum Platform Server
2. Open the **Spectrum Technology Platform** home page.
To do this, in click **Start > Precisely > Welcome Page > .**
3. Click **Platform Client Tools > Web > Open Management Console.**
4. Enter a valid user name and password in the **Sign in** dialog box.
5. Click **Resources > Connections** and create two connections.

Importing and Deploying Files to the Server

With connection in place, now you need a command line utility to import and deploy the files to the server.

1. Open the **Spectrum Technology Platform** home page.
2. Click **Platform Client Tools > Command Line** and download these files:
 - **Job Executor**, which is an individual *.jar file
 - **Administration Utility**, which is a zipped file (spectrum-cli)
3. Unzip the spectrum-cli file.
4. Copy all the files from the `Sample Template` folder to the `spectrum-cli` folder. This saves you from entering full path of every file with each command.
5. Run the `cli.cmd` utility located in the `spectrum-cli` folder to launch the command line interface.
6. Connect to the Spectrum Server with this command:

```
connect server:port --u username --p password
```

The connection is confirmed by this message: `Connected to server server name : port.`

7. Run the following commands in the order shown to import and deploy sample data flows:
 - a) Import modelstore using this command that imports the modelstore with all dependencies:

```
spectrum modelstore bulkImport --importDependency true
```

- b) Deploy modelstore using command:

```
modelstore deploy --n name
```

- c) Import dbconnection using command:

```
dbconnection import --f name.json
```

- d) Import data flows using command:

```
dataflow import --f name.df
```

Remove any extra spaces around commands of file names and replace the placeholder `<name>` with the exact name of the file.

Viewing Imported Files

After successful import, you can use these data flows to demonstrate the Spectrum capabilities.

Files are stored at following locations:

- **Modelstore:** In the Physical Model and Model Store tabs in Discovery

- **Dataflow:** In the Server Explorer of Enterprise Designer application
- **Json:** In the Model Store in Management Console

Integration between SugarCRM OnPremises and Microsoft Dynamics 365 Online

This section illustrates the migration of Account and Contacts from SugarCRM OnPremises to Microsoft Dynamics 365 Online systems. If there is an existing association of Contacts and Account in the SugarCRM, the migration takes care to maintain the same in Microsoft Dynamics 365 Online system as well.

The Sample Template contains following files:

Table 152: Files in Sample Template folder

File Type	File Name
Data Flow	<ul style="list-style-type: none"> • SugarCRMAccount_Sync_MSAccount.df • SugarCRMContact_Sync_MDSContact.df
DB Connection	<ul style="list-style-type: none"> • MSDynamics_MS.json • SugarCRM_MS.json
ModelStore	<ul style="list-style-type: none"> • mi_modelStore_MSDynamics_MS.smims • mi_modelStore_SugarCRM_MS.smims

Do the following to migrate data from SugarCRM OnPremise to MS Dynamics 365 Online

1. Create following connections:
 - a. **SugarCRM_TestConnection** with Type SugarCRM. Click Test and a success message is displayed: *The connection SugarCRM_OnPremises successfully connected to the data source.*
 - b. **MSDynamics_TestConnection** with Type Microsoft Dynamics 365. Click Test and a success message is displayed: *Success: The connection Microsoft Dynamics 365 Online successfully connected to the data source.*

- Import and deploy files using Spectrum Command line utility. Use following commands in given order:

- Import modelstore using command:

```
modelstore bulkimport --importDependency true
```

- Deploy modelstore using command:

```
modelstore deploy --modelStoreName MSDynamics_MS
modelstore deploy --modelStoreName SugarCRM_MS
```

- Import dbconnection using command:

```
dbconnection import --f MSDynamics_MS.json
dbconnection import --f SugarCRM_MS.json
```

- Import data flows using command:

```
dataflow import --f SugarCRMAccount_Sync_MSDAccount.df
dataflow import --f SugarCRMContact_Sync_MDSContact.df
```

- Launch the Enterprise Designer application. You can download the executable setup of this application from **Desktop** section of **Platform Client Tool** on the Spectrum Platform home page.
- Login using your Spectrum credentials.
- Click **View** from the menu; click **Server Explorer**.
- Double-click the **SugarCRMAccount_Sync_MSDAccount** dataflow job first.
- Double-click Read from **DB_SugarCRM** stage.
 - Change the value in the field **date_entered** as required and click **OK**

```
Select "SugarCRM_PM"."Accounts"."email1",
       "SugarCRM_PM"."Accounts"."name",
       "SugarCRM_PM"."Accounts"."phone_office",
       "SugarCRM_PM"."Accounts"."date_entered" From
       "SugarCRM_PM"."Accounts" Where
       "SugarCRM_PM"."Accounts"."date_entered" Like '2017-08-28%'
```

- Click **OK** to continue.

- Run the flow by clicking **Run** button on the Tool Bar in Enterprise Designer.
- Visit the MS Dynamics home page and click **Accounts** under **Customers** tab on the **Sales** page. Notice that the **Accounts** have been migrated.
- Double-click the **SugarCRMContact_Sync_MDSContacts** job.
- Double-click **Read from_MSDynamics_DB** stage.
 - Change the value of the field **createdon** as required and click **OK**

```
Select "MSDynamics_PM"."account"."name",
       "MSDynamics_PM"."account"."telephone1",
       "MSDynamics_PM"."account"."emailaddress1",
       "MSDynamics_PM"."account"."createdon",
```

```
"MSDynamics_PM"."account"."accountid" From
"MSDynamics_PM"."account"
Where "MSDynamics_PM"."account"."createdon" Like '2017-09-11%'
```

- b. Click **OK** to continue
12. Run the flow by clicking **Run** button on the Tool Bar in Enterprise Designer
13. Visit the MS Dynamics home page and click **Contacts** under **Customers** tab on the **Sales** page. Notice that the **Contacts** are listed on the page

The successful listing of accounts and contacts in the MS Dynamics page complete the migration process.

Integration between Salesforce and Oracle Eloqua

This section illustrates the migration of Account and Contacts from Salesforce to Oracle Eloqua systems. If there is an existing association of Contacts and Account in the Salesforce, the migration takes care to maintain the same in Oracle Eloqua system as well.

The sample template contains these files:

Table 153: Files in Sample Template folder

File Type	File Name
Data Flow	<ul style="list-style-type: none"> • <i>SalesforceAccount_Sync_OracleEloquaAccount.df</i> • <i>SalesforceContact_Sync_OracleEloquaContact.df</i>
DB Connection	<ul style="list-style-type: none"> • <i>Eloqua_MS.json</i> • <i>Salesforce_MS.json</i>
ModelStore	<ul style="list-style-type: none"> • <i>mi_modelStore_Eloqua_MS.smims</i> • <i>mi_modelStore_Salesforce_MS.smims</i>

To migrate data from Salesforce to Oracle Eloqua, perform these steps:

1. Create these connections:

- a. **Salesforce_TestConnection** with Type Salesforce. Click Test and a success message is displayed: *The connection Salesforce successfully connected to the data source.*
 - b. **Eloqua_TestConnection** with Type Oracle Eloqua. Click Test and a success message is displayed: *Success: The connection Eloqua successfully connected to the data source.*
2. Import and deploy files using the Spectrum Command line utility. Use following commands in this order:
 - Import modelstore using command:


```
modelstore bulkimport --importDependency true
```
 - Deploy modelstore using command:


```
modelstore deploy --modelStoreName Eloqua_MS
modelstore deploy --modelStoreName Salesforce_MS
```
 - Import dbconnection using command:


```
dbconnection import --f Eloqua_MS.json
dbconnection import --f Salesforce_MS.json
```
 - Import data flows using command:#


```
dataflow import --f SalesforceAccount_Sync_OracleEloquaAccount.df
dataflow import --f SalesforceContact_Sync_OracleEloquaContact.df
```
 3. Launch the Enterprise Designer application. You can download the executable setup of this application from **Desktop** section of **Platform Client Tool** on the Spectrum Platform home page.
 4. Login using your Spectrum credentials.
 5. Click **View** from the menu, click **Server Explorer**.
 6. Double-click the **SalesforceAccount_Sync_OracleEloquaAccount** dataflow job first.
 7. Double-click **Read from DB** stage.
 - a. Change the value of field CreatedDate as required and click **OK**.

```
Select "SalesforcePM"."Account".* From "SalesforcePM"."Account"
where CreatedDate > '2017-01-01'
```

- b. Click **OK** to continue.

8. Run the flow by clicking **Run** button on the Tool Bar in Enterprise Designer.
9. Visit the Eloqua home page and click **Account**. Notice the accounts have migrated to Eloqua.
10. Navigate back to Enterprise Designer.
11. Double-click the **SalesforceContact_Sync_OracleEloquaContact** job.
12. Double-click **Read from DB** stage.
 - a. Change the value of the field CreatedDate as required and click **OK**.

```
Select "SalesforcePM"."Contact".* From "SalesforcePM"."Contact"
where CreatedDate > '2017-01-01'
```

- b. Click **OK** to continue.
 13. Run the flow by clicking **Run** button on the Tool Bar in Enterprise Designer.
 14. Visit the Eloqua home page and click **Contacts**. Notice the contacts have migrated to Eloqua.
- The successful listing of accounts and contacts in the Oracle Eloqua complete the migration process.

Note: For more details on how to create connections, see the *Discovery Guide*.

8 - About Spectrum Technology Platform

In this section

- What Is Spectrum Technology Platform?.....878
- Enterprise Data Management Architecture.....879
- Spectrum Technology Platform Architecture.....883
- Modules and Components.....887



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 master 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.

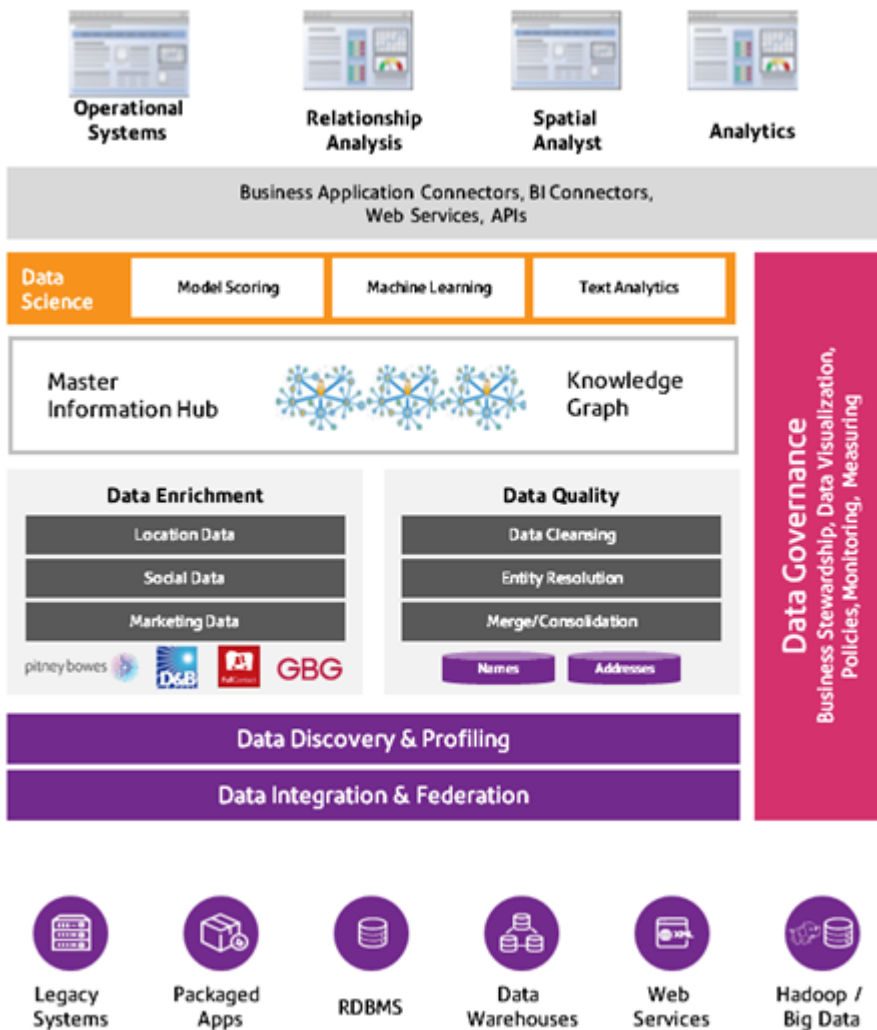
Spectrum Technology Platform software from Precisely integrates up-to-date jurisdictional boundaries with the exact street addresses of your customer records, enabling you to append the correct state, county, township, municipal, and special tax district information to your records. Some example uses 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 Information Hub

The Master Information Hub 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.

Spectrum Technology Platform supports two approaches to the MDM hub. In the master hub approach, the data is maintained in a single MDM database and applications access the data from the MDM database. In the registry approach, the data is maintained in each business application and the MDM hub registry contains keys which are used to find related records. For example, a customer's record may exist in an order entry database and a customer support database. The MDM registry would contain a single key which could be used to access the customer data in both places.

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 modules 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.

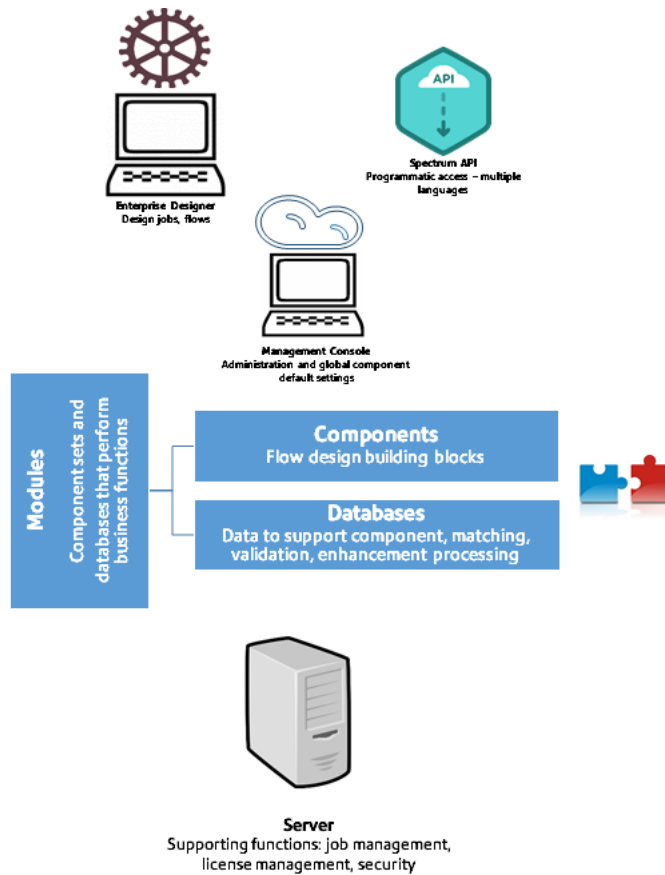
Spectrum Technology Platform can trigger batch processing based on the appearance of one or more source files in a specified folder. This "hot folder" trigger is useful for monitoring FTP uploads and processing them as they occur.

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 modules. These modules provide different 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.

Modules

Modules 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. Modules are grouped together to solve common business problems and licensed together as bundles.

Components

Modules are comprised of components which perform a specific function 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 modules 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.

Modules have both required and 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

Management Console | Flows | Services | Resources | System

Home > Resources: Connections

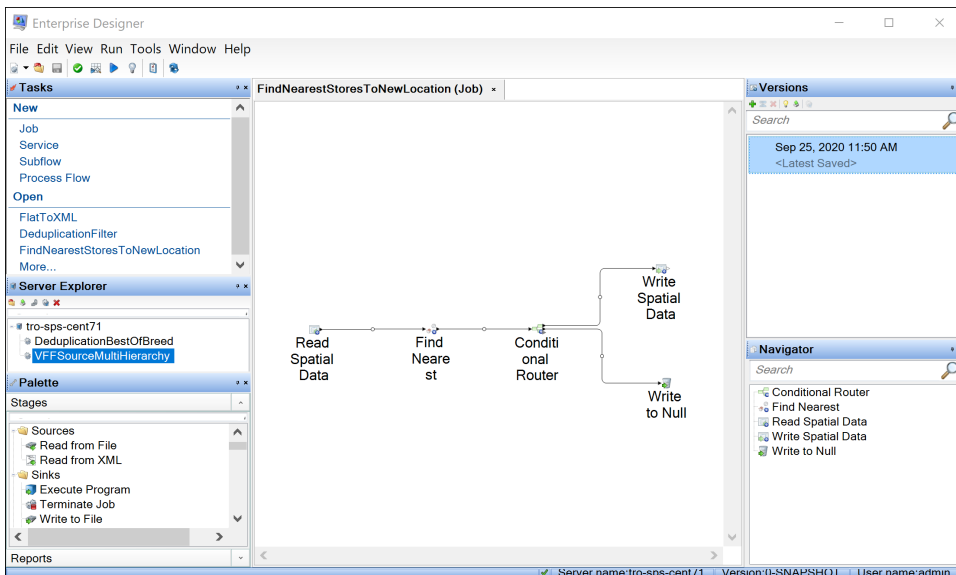
Connections

Connection Name	Connection Type
FTP	FTP
MS-SQL	MSSQLServer
Oracle	Oracle
Postgres	Postgres
ProfilerInternalConnection	Model Store

Showing 5 of 5 records Rows per page: 10

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.



Note: Spectrum Enterprise Designer will be replaced by Spectrum Flow Designer in a future release. Spectrum Flow Designer is in Technical Preview status at this time.

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 some component applications.

Modules and Components

Table 154: 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
Spectrum Screener	Helps banks and financial institutions to effectively detect financial crimes, reduce false positives, and maintain robust detection capability as required by the regulators.	Party Groups Lists Screen Alerts
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 Parser 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.	<ul style="list-style-type: none"> 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.	<ul style="list-style-type: none"> Get Route Data Get Travel Boundary Get Travel Cost Matrix Get Travel Directions Persistent Update
Spectrum SAP	Enables Spectrum Technology Platform to interface with SAP Customer Relationship Management applications.	<ul style="list-style-type: none"> 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.	<ul style="list-style-type: none"> Get Candidate Addresses Get City State Province Get Postal Codes Validate Address Validate Address Global

Module	Description	Components
Spectrum Universal Name	Parses personal names, company names, addresses, and many other terms and abbreviations.	Name Parser (Deprecated) Name Variant Finder Open Name Parser



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

www.precisely.com

© 2007, 2021 Precisely. All rights reserved.