

precisely

Spectrum Technology Platform

Flow Designer Guide

Version Technical Preview



Table of Contents

1 - Getting started

Browser behavior.....	4
Accessing Spectrum Flow Designer.....	4
Field data formats.....	5
Data Types.....	7
Flow Designer Home page.....	8
Explorer page.....	11
New Flow page.....	15
Flow Editor page.....	16
Profile page.....	20
Types of Flows.....	20
Starting a new Flow.....	22
Introduction to stages.....	23

2 - Components of Flows

Types of Flows.....	43
Starting a new Flow.....	44
Flow Input.....	45
Input stage types.....	47
Flow Output.....	50

3 - Building, testing, and running flows

Designing and building flows.....	54
Deleting flows.....	94
Exposing flows.....	95
Importing flows.....	95
Exporting flows.....	96
Running an External Program.....	98
Using the template browser to create flows and jobs.....	99

4 - Stages Reference

Flow Designer stages.....	103
Control.....	103
Sources.....	112
Sinks.....	113
Module Stages.....	115

5 - File properties reference

Supported character encoding methods.....	166
Field separators.....	166
Record separators.....	167

6 - Supported data types reference

Data types.....	169
-----------------	-----

7 - Date and time patterns reference

Date and time patterns.....	171
-----------------------------	-----

1 - Getting started

Flow Designer is a visual tool for creating work flows. Using this client, you can:

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

In this section

Browser behavior.....	4
Accessing Spectrum Flow Designer.....	4
Field data formats.....	5
Data Types.....	7
Flow Designer Home page.....	8
Explorer page.....	11
New Flow page.....	15
Flow Editor page.....	16
Profile page.....	20
Types of Flows.....	20
Starting a new Flow.....	22
Introduction to stages.....	23



Browser behavior

Flow designer supports current versions of Microsoft Edge (Chromium), Mozilla Firefox, and Google Chrome.

Note: Flow Designer opens new pages in new (separate) tabs.

A notable characteristic is that Firefox will open new tabs when opening a flow or when you select the **Home** button on the workflow editor page.

Note: Downloaded files are saved to a default location defined by your browser settings. Browser settings apply to all downloads and cannot be set specifically for Flow Designer.

Accessing Spectrum Flow Designer

Use your browser to access Spectrum Flow Designer from the Spectrum Technology Platform Welcome Page.

To start Spectrum Flow Designer:

1. Open a web browser and navigate to the Spectrum Technology Platform Welcome Page.

Enter the URL in the following format:

```
http://server:port
```

- *server* is the server name or IP address of your Spectrum Technology Platform server.
 - *port* is the HTTP port configured for the Spectrum Technology Platform server. The default for HTTP is 8080.
2. Click **Platform Client Tools**.
 3. Click the **Web** heading to expand the selections.
 4. Click **Open Flow Designer**.
 5. Enter a valid **User name** and **Password** and click **Sign in**.

Field data formats

Automatic Data Type Conversion

When your flow presents data to a stage for processing, but the data is not in an acceptable format (data type), Spectrum Technology Platform can, in some cases, automatically convert the data to allow processing.

Note: This feature will be available for all stages in a future release.

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 requires numeric data. If Math detects incoming string data, Spectrum Technology Platform can convert the data to the data as defined in the Math stage's **File** tab.

Automatic data type conversions occur 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:

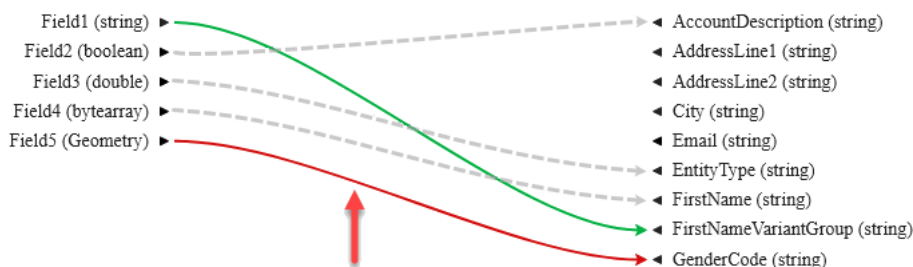


If you double-click the channel you can see the data type conversions for mapped fields. Data type conversions are identified by a dashed map channel. In the example below, the Open Name Parser output field **NameScore**, which is automatically mapped to the Write to File input field **NameScore**, and converted from integer to string format.



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 map channel.



You can specify the processing of conversion/mapping errors using type conversion options.

Related tasks

[Defining Data Type conversion for mapped fields](#) on page 72

Spectrum Technology Platform automatically changes field data types as needed using the type conversion settings specified in Management Console, or through the dataflow type conversion options specified in Flow Designer.

Reserved Field Names

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

- Status
- Status.Code
- Status.Description

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](#).

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

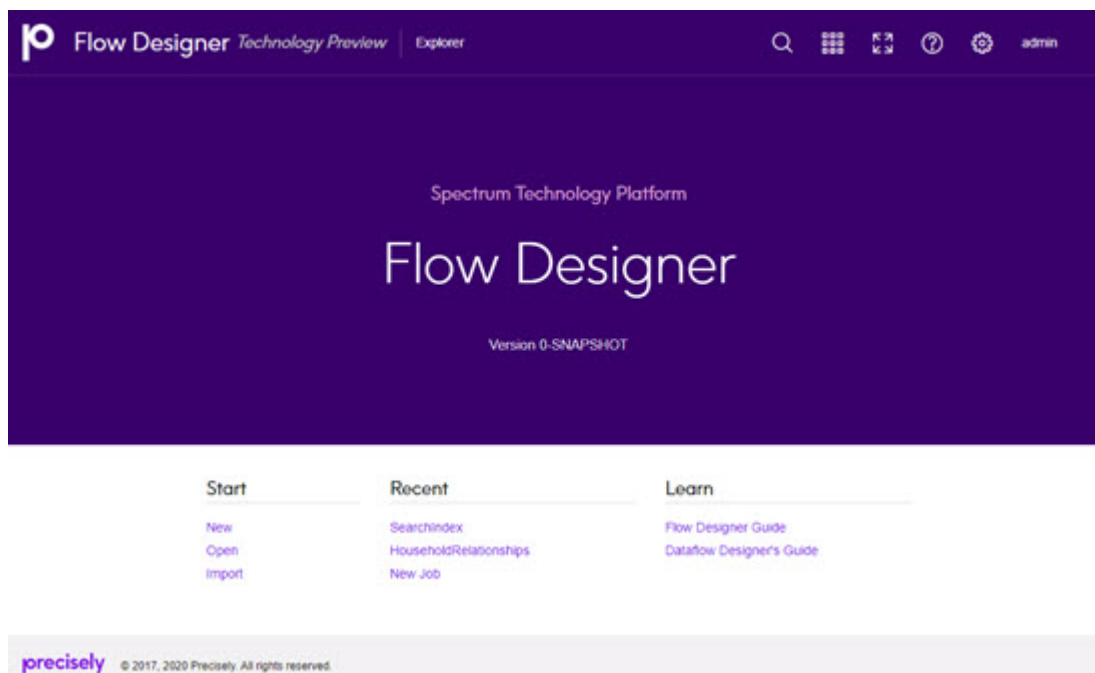
[Data types](#) on page 169

Spectrum Technology Platform supports these data types.

Flow Designer Home page

The **Home** page is the first page you see when you start Flow Designer.

This page is the initial access point for Flow Designer features.





Application toolbar - Home page

The **Application toolbar** is at the top right corner of the **Home** page.




The Application toolbar offers these tools and tasks:

Search tool


Click the Search tool  to open an input box that allows you to search for flows by name or by key string. Click the cancel search  button or press Esc to exit the Search tool.

Note: You can also use the keyboard shortcut Ctrl+F to search.


Application Switcher

Click the Application Switcher  to open another application in the same browser page.


Full-Screen Mode tool

Click the Enter full screen tool  to expand the display. Press Esc to exit full-screen mode.

Help tool

Click the Help  tool to access documentation for the current application or for all of spectrum, the keyboard shortcuts reference, or to contact Precisely support.

Application Setting tool

- Use the Application settings tool  to clear the history of recently opened flows (**Clear Recently Opened**).
- Click the user name (**admin** in the example) to configure profile settings (**Email, Password, Language**) or to **Sign out**.

Version information

The currently installed version is identified the middle of the Flow Designer Home page.

Start, Recent, and Learn access points

- Under **Start**:
 - Click **New** to display the **New Flow page** on page 15, in a new tab, which allows you to create a new empty flow or a new flow from a template.
 - Click **Open** to display the **Explorer page** on page 11, in a new tab, with a list of saved and exposed flows.
 - Click **Import** to display a file selector. Go to an existing data flow *.df file to open in Flow Designer.
- Under **Recent**, click any flow to open it on the **Flow Editor page** on page 16.
- Under **Learn**, click any help resource to open or access it.

The canvas

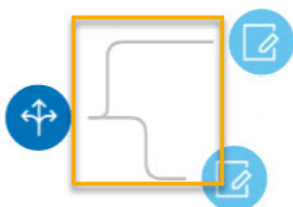
The canvas is the main work area, located on the **Explorer** page. To access the canvas, either click **Open** or on the **Home** page title bar click **Explorer**.

Using the canvas, you can:

- Edit saved flows.
- Save an existing flow with a new name to create a starting point for a new flow.

Learn more about the controls on the Manager page and its controls: [Explorer page](#) on page 11.

Channel A channel is a connection between two or more stages through which records are passed from one stage to another.



Port If you look closely at the stage icons you will notice small round port indicators on one or more sides of a stage. A port sends data to stages or reads data from a channel. Stages that read data into the flow ("sources") have output ports since they are at the start of a flow. Stages that send data out of the flow ("sinks") have input ports since they are at the end of a flow. All other stages have both input and output ports.



Read from File

Some stages have error ports, which output records that cause errors during the stage's processing. Some stages have report ports, which generate reports about the stage's output.

You can hover over a port indicator to view its name.



You can double-click a port indicator to open a page that allows you to configure that port's details. The configuration settings vary depending on the type of operation being defined. Ports show different colors based on their status. The colors, in order of priority are:

- pink - exception error
- sky blue - report
- green - input or output success
- gray - normal status

List and work with saved flows

Flow Designer displays a list of saved flows, providing quick and easy access to those flows.

To open the list view: From the **Flow Designer Home page** on page 8, click **Open** or on the title bar click **Explorer** on the title bar. In the list view, you can do any of the following:

- View details** See the locked status, flow type, exposed version, and last date and time the flow was modified.
- Open a flow** Click on any name to open the flow.
- Select multiple flows** Check any box to edit, delete, export, or unlock a locked flow.
- Global search** Specify a keyword in the entry field. Lists any flow name or type that contains the string you enter. Click **X** to clear search criteria.
- Sort** Use the up and down sort arrows next to any column heading to sort on that field. Or, click on the column heading to toggle the sort order.

Explorer page

The **Explorer** page displays and provides access to flows. To open the **Explorer** page from the **Home** page, either click **Open** or on the title bar click **Explorer**.

The screenshot shows the Flow Designer Explorer page. At the top, there is a navigation bar with the Flow Designer logo and 'Technology Preview' text. Below the navigation bar, the breadcrumb 'Home > Explorer' is visible. The main content area is titled 'Explorer' and contains a table of saved flows. The table has the following columns: Name, Locked By, Type, Exposed Version, and Last Modified. The flows listed are HouseholdRelationships, SearchIndex, and TerrorUseCase. Below the table, there is a pagination control showing '1-3 of 3' and a 'To import your dataflows, drag and drop your .df files here.' message. The footer contains the Precisely logo and copyright information: '© 2017, 2020 Precisely. All rights reserved.'

Name	Locked By	Type	Exposed Version	Last Modified
HouseholdRelationships	admin	Job		9/25/20, 3:54 PM
SearchIndex	admin	Job		9/25/20, 3:54 PM
TerrorUseCase	admin	Job	Imported Tue Sep 15 19:41:29 EDT 2020	9/15/20, 7:46 PM

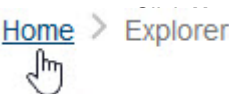
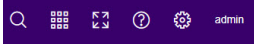





From this page you can:

- Edit saved flows.
- Import or export saved flows.
- Save an existing flow with a new name to create a starting point for a new flow.
- Return to the Home page.

Icons, toolbar, and controls - Explorer page

This topic provides a reference to the controls on the **Explorer** page.

Table 1: Controls on the Explorer page

Home link		Click the Home link in the navigation path below the Application toolbar to return to the Home page.
Application toolbar		<p>The Application toolbar is at the top right corner of the page and supports these tools and tasks:</p> <ul style="list-style-type: none"> • Search • Application menu • Enter full screen • Help • Application setting • Display Version information • Provide Start, Recent, and Learn access points
Search for a flow by name		<p>Use the Search tool to open an input box that allows you to search for flows by name or by key string.</p> <p>Tip: You can also press Ctrl+F to search.</p>
Application menu		Use the Application menu to start another application in the same browser page.
Enter full screen		Click the Enter full screen button to expand the display. Press ESC to exit full-screen mode.
Help button		Click the Help button to access documentation, access the keyboard shortcuts reference, or to contact Precisely support.
Application setting tool		<ul style="list-style-type: none"> • Use the Application setting tool to clear the history of recently opened flows. • Click the user name (in our example, "admin") to change user profile settings or to sign out.

Version information Find the installed version information in the middle of the Flow Designer Home page.

Start, Recent, and Learn access points






- Under **Start** click one of the following:
 - **New** to display the New Flow page, in a new tab, which allows you to create a new empty flow or a new flow from a template.
 - **Open** to display the **Explorer** page, in a new tab, with a list of saved and exposed flows.
 - **Import** to display a file selector. Go to an existing data flow *.df file to open in Flow Designer.
- Under **Recent**, click any flow to open it on the Edit Job page.
- Under **Learn**, click any help resource. Under **Learn**, click any help resource to open or access it.

Paging controls - Explorer page

This topic provides a reference to the paging and display controls on the **Explorer** page. The **Explorer** page has a group of buttons that let you change the content and number of flows displayed in the table.

The **Explorer** page has a group of buttons that let you change the content and number of flows displayed.

Table 2: Controls - Explorer page

	Scroll to the top of the list
	Scroll up one page
	Select a page to display from the series of pages. The highlighted page is the currently displayed page in the series.
	Scroll down one page
	Scroll to the bottom of the list











Click the drop-down menu to specify the number of rows display in the flow list. If the number of flows exceeds the number of rows defined, Flow Designer shows the page count in the series of pages display.

Task buttons - Explorer page

This topic provides a reference to the Task buttons on the **Explorer** page.

The **Explorer** page task buttons support tasks that allow you to work with new and existing flows.



New flow		Opens the New Flow dialog, allowing you to select a template for a new flow, or to open the blank job workspace to define a new flow on a blank canvas.
Edit selected flow		Edit the selected flow using the canvas and its available tools and sources. Edit opens one or more selected flows in the flow editor.
Lock or unlock selected flow		When you lock flows, you prevent others from making changes.
Delete flows		Deletes the selected flow.
Import flows		Imports flows that were previously saved to dataflow (.df) files.
Export flows		Exports selected flows to dataflow (.df) files.
Copy selected flows		Copying is useful for applying existing elements to create a new flow.
Refresh the list of flows		Refresh the flow list after you make changes. Click this button, periodically, to capture changes submitted by other users in a shared environment.

Related tasks

Deleting flows on page 94

This topic provides the steps for deleting flows and describes the confirmation messages returned by the delete function.

Exporting a selected flow

Importing flows - import button method on page 95

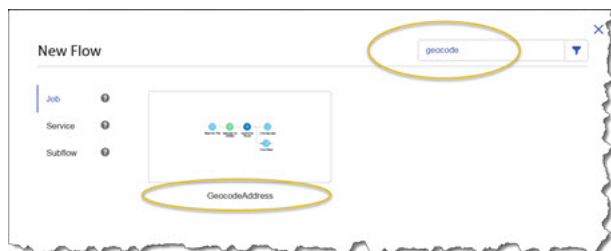
New Flow page

The **New Flow** page is the starting point for building new flows. Flow Designer provides a template browser that allows you to select templates to use in creating new flows.

To access this page, click **New** on the **Home** page. Alternatively, click the New button **+** on the **Explorer** page.

Filter and find

Use the Filter input box to quickly find a template based on a keyword.



Show templates

Click **Job**, **Service**, or **Subflow** to display the corresponding templates. Click on any help tool **?** next to any template label to display a description of the types of flows.

Inspect templates

Hover over any template, and you will see the Zoom tool **🔍**. Click Zoom in **🔍** to inspect template content before opening a template. Click Zoom out **🔍** to minimize the template.

Related concepts

Flow templates on page 99

Flow Editor page

The **Flow Editor** page contains a canvas with a palette and workflow task buttons that you use to modify existing flows.


To access the **Flow Editor** page, select the check box next to a flow on the **Explorer** page, and click the Edit button  button. On the **Flow Editor** page you can:

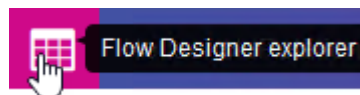
- Work with a new or existing flow
- Import or export flows
- Perform last actions
- Edit flow details
- Run a flow
- Change the display


Toolbar and controls (Edit Job page)

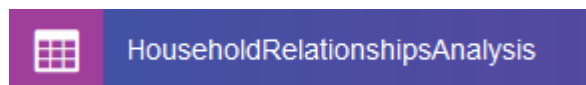
Home button and flow name

These items display at the top left corner of the page and support these tasks:

Return to the Explorer page button 



The current flow name appears next to the Return to Explorer button 



Application toolbar

The **Application toolbar** is at the top right corner of the page and supports these tasks:

Enter full screen mode



Access help resources: View documentation, View the keyboard shortcuts reference, or contact Precisely support.



Sign out of Flow Designer (click **user name** > **Signout**)



Canvas task buttons

The canvas task buttons allow you to interact with the workflow space.

Open the New Flow page to select a flow, job, or service to add to the flow currently displayed on the canvas.



Open the **Explorer** page, and display the flow list to select and open one of the available flows.

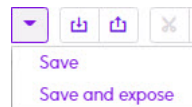


Save the flow displayed on the canvas.



Select a Save method:

- **Save** - Perform a standard save.
- **Save and expose** - Save the flow, creating a versioned copy.



Note: All saved flows are visible (exposed) to other Flow Designer users.

Import a flow



Export a flow



Undo the last action (maximum 100)



Repeat the last action



Cut the selected item from the canvas



Copy the selected item to the clipboard



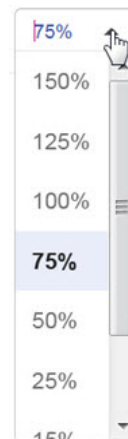
Paste the item on the clipboard to the canvas



Permanently delete the selected item



Select the workflow canvas display size as a percentage of the actual size



Fit the canvas display to the page (screen)



Run the flow that is currently displayed



Toggle the grid line display



Toggle overview display: Click to open a small overview display that shows you the current focus of a workflow display. This is useful for larger, complicated workflows.



Refresh the current display



Close the flow: This prompts to save changes.



Panel display and view settings

These controls change the display of the palette panel.

Show the Validation Errors panel or Hide the Validation Errors panel. This panel evaluates the accuracy of the flow on the canvas and reports the errors in a concise view. The errors will contain the stage and a description of the error.



Show the Inspection panel or Hide the Inspection panel. This panel provides a brief but expandable view of the selected items.



Show the Execution History panel or hide the Execution History panel. This panel provides a quick way for you to run flows and obtain processing results.



Filter

Immediately below the palette view settings is the filter entry box. You can narrow the stages displayed by entering a string, such as "read," and see only the stages containing that description.



Palette display and view settings

The palette has a robust set of display settings that you can change as needed.

Toggle to hide or show the palette panel (**CTRL+C**)



List mode display - lists all of the stages available in a compact view with small icons. With this view, you can quickly scan available stages.



Icon mode – lists all of the stages with larger icons. Since this display emphasizes icons, more scrolling is required to see all the stages in a grouping.



Expand all – shows all of the stages in all groupings as indicated by the selected (List or Icon) display mode.



Collapse all – Shows a list of available Sources, Sinks, and Control stages, deployable modules, and user-defined stages in a list format. Click on any label to expand the display and see the stages under each label as indicated by the selected (List or Icon) display mode.



Stages

The palette shows the stages you can apply to your flows.

Related concepts

[Introduction to stages](#) on page 23

The palette on the **Flow Editor** page, adjacent to the canvas, shows the stages you can apply to your flows.

Profile page

The **Profile** page allows you to configure settings for the current user. To access this page, click the user name on the application toolbar on either the **Home** or the **Explorer** page, then click **Email**, **Password**, **Language**.

The **Profile** page contains these fields and selections:

User name	The user name comes from the current user login credentials. You cannot edit this field.
Email	Use this input field to specify an email address for the current user.
Change password	Click to open Change Password dialog box to enter a new password.
Language	Select one of the supported languages from the Language drop-down, and the user interface will change the language display settings.
Country	This specifies the locale to narrow the dialect for the specified language. For example, if you choose Español as your language, you can select a specific Spanish-speaking locale, such as Chile, to localize the dialect.

Types of Flows

A flow is a series of operations that takes data from some source, processes that data, then writes the output to a destination. The processing of the data can be take the form of simple sorting to more complex data quality and enrichment actions.

While the concept of a flow is simple, you can design very complex flows with branching paths, multiple sources of input, and multiple output destinations. The [New Flow page](#) on page 15 is the starting point to build a job flow, service flow, or subflow.

Job

A job is a flow 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. Run jobs manually in Flow Designer or from a command line using the job executor.

The job example, below, uses the Read from File stage for input and two Write to File stages as output.



Service

A service is a flow 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 module. For example, when you install the Universal Addressing Module the service ValidateAddress becomes available on your system. In other cases, you must create a service in Flow Designer, then expose that service on your system as a user-defined service. For example, Spectrum Spatial stages are not available as services unless you first create a service using the module's stages.

You can also design your own custom services. For example, you can create a flow that helps to determine whether an address is at risk for flooding.

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

Subflow

A subflow is a flow that can be reused within other flows. Subflows are useful when you want to create a reusable process that can be easily incorporated into flows. 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.

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

In this type of flow, 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 flow, in this case Geocode US Address.

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.

Starting a new Flow

To start a new **Flow**, use these steps:

1. On the **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** that you are creating, and click **OK**. The <name of the flow> page is displayed.
5. From the **Palette Panel**, drag the required stage to the canvas.
6. Use the **Input Port** to define inputs to the stage.

Note: For details on defining inputs, see [Flow Input](#) on page 45.

7. Use the **Output Port** to define outputs to the stage.

Note: For details on defining output, see [Flow Output](#) on page 32

8. To configure the stage options and settings, click on the stage.

Note: See the respective stage documentation for these settings.

Introduction to stages

The palette on the **Flow Editor** page, adjacent to the canvas, shows the stages you can apply to your flows.

Note: The stages and sinks displayed in the **Stages** palette are grouped and color-coded by type.

Sources	Define the input for a flow. A source is the first stage in a flow, and defines the input data to process.
Sinks	Define the output from a flow. A sink is the last stage in a flow, and 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.
Control stages	Move data along different paths in a flow, to split or group records, and to perform basic data transforms and mathematical operations. Custom transforms, a type of control stage, are supported in this preview release of Flow Designer. You also have the option of configuring custom transforms through Enterprise Designer for use in your Flow Designer jobs.
Module-specific stages	Provide functions that are available through the modules you have installed on the Spectrum Technology Platform. Note: We will add module-specific stages in a future release.
User-defined stages	Provide a means to develop custom operation stages for your flows.

Input and Read from File stages

The input stage type for a flow depends on the type of flow.

Input stage types

The input stage type for a flow depends on the type of flow. 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.

- Jobs use the Read from File input stage. For more information, see [Configure the Read from File Stage](#) on page 54.
- Services use the Input stage. For more information, see [Configure the Input stage](#) on page 56.
- Subflows use the Read from File, Input, or Read from XML stage since they accept input from Jobs and Services. For more information, see [Configure the Read from XML Stage for Subflows](#) on page 59.

When you specify a flow's input file, Flow Designer examines the input file to determine as much of the file information as possible using that information to populate the fields on this page. The information it returns includes:

- Record type:
 - Line Sequential - 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.
 - Fixed Width - Each record is a specific number of characters in length and each field has a fixed starting and ending character position.
 - Delimited - 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.
- Character Encoding ([Supported character encoding methods](#) on page 166)
- Field Separator ([Field separators](#) on page 166)
- Text Qualifier – single or double quotes
- Record Separator ([Record separators](#) on page 167)
- File Schema (layout)

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 modules you have licensed. The Enterprise Data Integration Module provides access to the most data sources of any module.

Note: When designing a job, you should anticipate 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 25.

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.

See the solution guide for your modules available at support.precisely.com.

Related reference

Sources

File tab

The file tab defines relevant content and format information.

- Display details about the file's format and layout.
- Define the presence of a header record.
- Mark records with fewer fields than defined as "malformed."
- Use the convenient File Schema workspace to change record details using the user interface.

Related tasks

[Configure the File tab details - Read from File](#) on page 54

[Managing malformed input records](#) on page 25

Sort tab

The Sort tab describes a flow's sort key fields and sort order.

- Accept, delete, or change the sort fields used.
- Accept or change the order of the sort fields.
- Accept or change the sort direction for any sort field: ascending or descending.

See also: [Configuring the Sort tab](#)

Runtime tab

The Runtime tab defines a flow's input file processing options.

- Define the starting record to process.
- Define whether to process all input records or specify a maximum number of records to process.

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.

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.

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.

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

Use the Sorter stage to sort records using fields you specify.

You can configure these details for a Sorter stage:

- Order – Displays the sort fields in sort sequence
- Sort Field – Name of the sort field
- Direction – Sort direction: Ascending or Descending
- Sort Type – The type of content in the sort field, such as integer, string, and others
- Trim Blanks – Indicate whether to remove blank space before or after the field content
- Treat Nulls as – Select smallest or largest to direct the placement of null values in the sorted list
- Advanced – Advanced sort options that define overrides for sort performance. We suggest that you not change advanced sort performance options without consulting your system administrator.

Related concepts

[How Flow Designer sorts data](#) on page 31

Flow Designer applies the ASCII standard to sort fields, strings, and characters into ascending and descending order.

Related tasks

[Configure the Sorter stage](#) on page 77

These configuration steps apply to the Technical Preview version of Flow Designer. We will update the documentation for this stage when the full version of Flow Designer is available.

How Flow Designer sorts data

Flow Designer applies the ASCII standard to sort fields, strings, and characters into ascending and descending order.

Ascending order

The smallest or first or earliest items appear at the top of the list.

- Numbers or amounts: Sorts smallest to largest; lower numbers or amounts are at the top of the list.
- Letters or strings: From A to Z
- Combination strings, such as address lines: 0-9 then A-Z

Dates: The oldest dates are at the top of the list.

Descending order

The largest or last items appear at the top of the list:

- Numbers or amounts: Sorts largest to smallest. Higher numbers or amounts are at the top of the list.
- Letters or strings: From Z to A

- Combination strings, such as address lines: Z-A then 9-0
- Dates: Most recent dates are at the top of the list.

Sort order precedence

This chart (*source: ecisolutions.com*) shows characters processed in ascending order. Reverse the characters to determine the order for descending sorts.

space	8	P	h
!	9	Q	i
"	:	R	j
#	:	S	k
\$	<	T	l
%	=	U	m
&	>	V	n
' (apostrophe)	?	W	o
(@	X	p
)	A	Y	q
*	B	Z	r
+	C	[s
, (comma)	D	\	t
- (dash)	E]	u
. (period)	F	^	v
/	G	_(underline)	w
0	H	^(ticmark)	x
1	I	a	y
2	J	b	z
3	K	c	}
4	L	d	
5	M	e	{
6	N	f	~
7	O	g	DEL

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.

Output details

Output or "Write to" stages define the output fields that a flow returns.

When you specify an output file for your flow, Flow Designer examines the incoming records to determine as much as it can about the output information, populating the File tab/page. The information it returns includes:

- [Output] File Name
- Record type
 - Line Sequential - 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.
 - Fixed Width - Each record is a specific number of characters in length and each field has a fixed starting and ending character position.
 - Delimited - 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.
- Character Encoding - For more information, see [Supported character encoding methods](#) on page 166.
- Field Separator - For more information, see [Field separators](#) on page 166 for a list of valid characters.
- Text Qualifier - Use single or double quotes.

- Record Separator - For more information, see [Record separators](#) on page 167 for a list of valid formats.

The Write to File stage configuration has three tabs: File, Sort, and Runtime.

File tab

The file tab defines relevant content and format information for your output.

Using the File tab, you can:

- Display details about the output file's format and layout.
- Define the presence of a header record.
- Mark records with fewer fields than defined as "malformed."
- Use the convenient File Schema workspace to change record details in the user interface.

Sort tab

The Sort tab describes a flow's sort key fields and sort order.

Using the Sort tab, you can:

- Accept, delete, or change the sort fields used.
- Accept or change the order of the sort fields.
- Accept or change the sort direction for any sort field: ascending or descending.

Runtime tab

The Runtime tab defines a flow's input file processing options.

With the Runtime tab, you can:

- Define the starting record to process.
- Define whether to process all input records or specify a maximum number of records to process.

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-bbox="488 852 1422 915">data['FieldName']</pre> <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 1020 1422 1083">data['CurrentBalance']</pre>
To set the value of a dataflow field	<p>Enter this code in the script editor:</p> <pre data-bbox="488 1230 1422 1293">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 1398 1422 1440">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>

Task	Instructions																																												
<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="487 441 1421 535">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="487 640 1421 766">int recordNumber = 1; data['Record_Number']= recordNumber; recordNumber++;</pre> <p>The output will be:</p> <table border="1" data-bbox="487 840 860 1165"> <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> <p>To change the scope of this variable to all input records, specify this:</p> <pre data-bbox="487 1249 1421 1396">import groovy.transform.Field @Field int recordNumber = 1; data['Record_Number']= recordNumber; recordNumber++;</pre> <p>The output will be:</p> <table border="1" data-bbox="487 1470 860 1795"> <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>	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																																												

Task	Instructions
<p>To create a new field using a numeric data type</p>	<p>Enter this code in the script editor:</p> <pre data-bbox="500 386 1425 449">data['FieldName'] = new constructor;</pre> <p>Where <i>constructor</i> is one of these:</p> <p>java.lang.Double(<i>number</i>) Creates a field with a data type of Double.</p> <p>java.lang.Float(<i>number</i>) Creates a field with a data type of Float.</p> <p>java.lang.Integer(<i>number</i>) Creates a field with a data type of Integer. You can also create a new integer field by specifying a whole number. For example, this will create an integer field with a value of 23:</p> <pre data-bbox="669 919 1367 982">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="487 1192 1425 1224">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="500 1369 1425 1432">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="669 1642 1367 1705">2013,12,23</pre> <p>com.pb.spectrum.api.datetime.Time(<i>hour,minute,second</i>)</p>

Task	Instructions
	<p>Creates a field with a data type of time. For example, 4:15 PM would be:</p> <pre>16,15,0</pre> <p>.</p> <p>com.pb.spectrum.api.datetime.DateTime(year,month,day,hour,minute,second)</p> <p>Creates a field with a data type of DateTime. For example, 4:15 PM on December 23, 2013 would be:</p> <pre>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['TransactionDate'] = new com.pb.spectrum.api.datetime.Date(2013,12,23);</pre>
<p>To create a new field with a data type of Boolean</p>	<p>Enter this code in the script editor:</p> <pre>data['FieldName'] = true or false;</pre> <p>For example, to create a field named IsValidated and set it to false, you would specify this:</p> <pre>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>NewListField = [] NewRecord = factory.create() NewRecord['NewField1'] = "Value" NewRecord['NewField12'] = "Value" ... NewListField << NewRecord NewRecord = factory.create() NewRecord['NewField1'] = "Value" NewRecord['NewField12'] = "Value" ...</pre>

Task	Instructions
	<pre data-bbox="500 317 1138 380">NewListField << NewRecord data['ListOfRecords'] = NewListField</pre> <p data-bbox="488 415 1424 485">For example, this creates a new list field called "addresses" consisting of two "address" records.</p> <pre data-bbox="500 520 1187 856">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> <p data-bbox="488 892 1424 999">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="500 1035 1117 1161">phoneNumbers = [] phoneNumbers << data['HomePhone'] phoneNumbers << data['WorkPhone'] data['PhoneNumbers'] = phoneNumbers</pre>
<p data-bbox="220 1262 435 1331">To concatenate fields</p>	<p data-bbox="488 1262 1424 1331">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="500 1367 1256 1461">String fullname = data['FirstName'] + ' ' + data['LastName']; data['FullName']=fullname;</pre> <p data-bbox="488 1497 1424 1566">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="500 1602 1219 1696">address1 = data['AddressLine1']; address2 = data['AddressLine2']; data['Address']=address1+ ', ' + address2;</pre>
<p data-bbox="220 1797 435 1829">To parse a field</p>	<p data-bbox="488 1797 1424 1866">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</p>

Task	Instructions
	<p>separates the five-character ZIP Code and the +4 portion and writes them to separate fields in the output record.</p> <pre data-bbox="488 401 1425 800"> 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>
<p>To perform conditional processing</p>	<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 data-bbox="488 1066 1425 1220"> city = data['City']; address1 = data['AddressLine1'] if(city.equals('Austin')) data['AddressCity']=address1 + ',' + city; </pre>
<p>To perform looping</p>	<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>
<p>To augment data</p>	<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 data-bbox="488 1556 1425 1709"> firmname = data['FirmName']; constant = 'Incorporated'; if(firmname.length() > 0) data['FirmName']=firmname + ' ' + constant; </pre>

Task	Instructions
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 data-bbox="488 422 1422 485">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 data-bbox="488 590 1422 653">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.
11. Click **OK**.

2 - Components of Flows

In this section

- Types of Flows.....43
- Starting a new Flow.....44
- Flow Input.....45
- Input stage types.....47
- Flow Output.....50



Types of Flows

A flow is a series of operations that takes data from some source, processes that data, then writes the output to a destination. The processing of the data can be take the form of simple sorting to more complex data quality and enrichment actions.

While the concept of a flow is simple, you can design very complex flows with branching paths, multiple sources of input, and multiple output destinations. The [New Flow page](#) on page 15 is the starting point to build a job flow, service flow, or subflow.

Job

A job is a flow 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. Run jobs manually in Flow Designer or from a command line using the job executor.

The job example, below, uses the Read from File stage for input and two Write to File stages as output.



Service

A service is a flow 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 module. For example, when you install the Universal Addressing Module the service ValidateAddress becomes available on your system. In other cases, you must create a service in Flow Designer, then expose that service on your system as a user-defined service. For example, Spectrum Spatial stages are not available as services unless you first create a service using the module's stages.

You can also design your own custom services. For example, you can create a flow that helps to determine whether an address is at risk for flooding.

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

Subflow

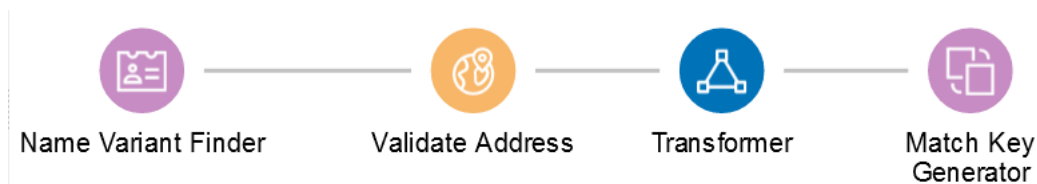
A subflow is a flow that can be reused within other flows. Subflows are useful when you want to create a reusable process that can be easily incorporated into flows. For example, you might want to create a subflow that performs deduplication using certain settings in each stage so that you can use the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

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

In this type of flow, 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 flow, in this case Geocode US Address.

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.

Starting a new Flow

To start a new **Flow**, use these steps:

1. On the **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** that you are creating, and click **OK**.

The <name of the flow> page is displayed.

5. From the **Palette Panel**, drag the required stage to the canvas.
6. Use the **Input Port** to define inputs to the stage.

Note: For details on defining inputs, see [Flow Input](#) on page 45.

7. Use the **Output Port** to define outputs to the stage.

Note: For details on defining output, see [Flow Output](#) on page 32

8. To configure the stage options and settings, click on the stage.

Note: See the respective stage documentation for these settings.

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 25.

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.

Input stage types

The input stage type for a flow depends on the type of flow. 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.

- Jobs use the Read from File input stage. For more information, see [Configure the Read from File Stage](#) on page 54.
- Services use the Input stage. For more information, see [Configure the Input stage](#) on page 56.
- Subflows use the Read from File, Input, or Read from XML stage since they accept input from Jobs and Services. For more information, see [Configure the Read from XML Stage for Subflows](#) on page 59.

When you specify a flow's input file, Flow Designer examines the input file to determine as much of the file information as possible using that information to populate the fields on this page. The information it returns includes:

- Record type:
 - Line Sequential - 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.
 - Fixed Width - Each record is a specific number of characters in length and each field has a fixed starting and ending character position.
 - Delimited - 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.
- Character Encoding ([Supported character encoding methods](#) on page 166)
- Field Separator ([Field separators](#) on page 166)
- Text Qualifier – single or double quotes
- Record Separator ([Record separators](#) on page 167)
- File Schema (layout)

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 modules you have licensed. The Enterprise Data Integration Module provides access to the most data sources of any module.

Note: When designing a job, you should anticipate 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 25.

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.

See the solution guide for your modules available at support.precisely.com.

Related reference

Sources

File tab

The file tab defines relevant content and format information.

- Display details about the file's format and layout.
- Define the presence of a header record.
- Mark records with fewer fields than defined as "malformed."
- Use the convenient File Schema workspace to change record details using the user interface.

Related tasks

[Configure the File tab details - Read from File](#) on page 54

[Managing malformed input records](#) on page 25

Sort tab

The Sort tab describes a flow's sort key fields and sort order.

- Accept, delete, or change the sort fields used.
- Accept or change the order of the sort fields.
- Accept or change the sort direction for any sort field: ascending or descending.

See also: [Configuring the Sort tab](#)

Runtime tab

The Runtime tab defines a flow's input file processing options.

- Define the starting record to process.
- Define whether to process all input records or specify a maximum number of records to process.

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.

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.

Output details

Output or "Write to" stages define the output fields that a flow returns.

When you specify an output file for your flow, Flow Designer examines the incoming records to determine as much as it can about the output information, populating the File tab/page. The information it returns includes:

- [Output] File Name
- Record type
 - Line Sequential - 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.
 - Fixed Width - Each record is a specific number of characters in length and each field has a fixed starting and ending character position.
 - Delimited - 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.
- Character Encoding - For more information, see [Supported character encoding methods](#) on page 166.
- Field Separator - For more information, see [Field separators](#) on page 166 for a list of valid characters.
- Text Qualifier - Use single or double quotes.
- Record Separator - For more information, see [Record separators](#) on page 167 for a list of valid formats.

The Write to File stage configuration has three tabs: File, Sort, and Runtime.

File tab

The file tab defines relevant content and format information for your output.

Using the File tab, you can:

- Display details about the output file's format and layout.
- Define the presence of a header record.
- Mark records with fewer fields than defined as "malformed."
- Use the convenient File Schema workspace to change record details in the user interface.

Sort tab

The Sort tab describes a flow's sort key fields and sort order.

Using the Sort tab, you can:

- Accept, delete, or change the sort fields used.
- Accept or change the order of the sort fields.

- Accept or change the sort direction for any sort field: ascending or descending.

Runtime tab

The Runtime tab defines a flow's input file processing options.

With the Runtime tab, you can:

- Define the starting record to process.
- Define whether to process all input records or specify a maximum number of records to process.

3 - Building, testing, and running flows

In this section

Designing and building flows.....	54
Deleting flows.....	94
Exposing flows.....	95
Importing flows.....	95
Exporting flows.....	96
Running an External Program.....	98
Using the template browser to create flows and jobs.....	99



Designing and building flows

Configure input stages

Configure the Read from File Stage

Follow these steps to configure the Read from File stage.

1. Open an existing flow or start with a blank workspace.
2. Select an Input stage from the palette and drag the stage to the canvas.
3. Double-click the input stage to open the configuration view. Note that the input configuration has three tabs: **File**, **Sort**, and **Runtime**.
4. Select an input file for this stage.

You are ready to [Configure the File tab details - Read from File](#) on page 54.

Configure the File tab details - Read from File

1. At the **File Name** field, click the file name button to display the **Choose File** selection dialog. Flow Designer displays a list of available files at the default location.
2. To select an input file, you can:
 - Go to another location using the navigation character (<).
 - Filter the list of files using a letter or string then clicking the filter button.
 - Click a to select a file name in the scroll box, and click **OK** to confirm your choice.

Flow Designer returns to the Read From File configuration page File tab, displaying your input file in the File **Name** field.

3. Review the populated File tab fields to ensure correctness and make adjustments if needed.
4. Check the boxes that provide more detail on the file content:
 - First row is header record – Toggle to **YES** if the first row in the input file is a header record.
 - Treat records with fewer fields than defined as malformed – Toggle to **YES** to implement this global record handling option. You can learn more about handling malformed records here: [Managing malformed input records](#) on page 25
5. Use the **File Schema** preview workspace to:
 - a) Change the order of the input fields:
 1. Click any gray bar in front of a field name; the pointer changes to a hand.

2. Hold down the mouse button and drag the field selection to its new position.
 3. Release the mouse button when the new position is correct.
 4. Click **Regenerate** to save the change.
- b) Rename a field – Double-click a field name to open a text input box, allowing you to change the name.
 - c) Change the field content type – Double-click the **Type** column for any field to display the drop-down, and select a valid format for the field content.
 - d) Selectively trim fields to their used length.
 - e) Selectively define a field's locale or format – If available, double-click the **Locale/Format** column for any field, and change the definition.
 - f) Selectively delete fields – Click the trash can: You will see a confirmation message before you can proceed.

Tip: Use the **Regenerate** button to re-examine and re-display the file schema after you make changes.


6. Click **Apply** so that your changes take effect, or click **Cancel** to return to the workflow canvas without saving changes.

You are ready to [Configure the Sort tab details - Read from File](#) on page 55.

Configure the Sort tab details - Read from File

The default view for the Sort tab assumes that your flow's sort order will copy the order in which those fields appear in the input file. You can change the defaults on this page.

Tip: You can click **Cancel** to abandon changes at any time.

1. Change the order of the sort fields.
For more information, see [Configure the File tab details - Read from File](#) on page 54.
2. Change the sort fields used.
 - a. Double-click the name of any sort field. Alternatively, click the field name once to select it and click the drop-down arrow to show a list of available sort fields.
 - b. Select a new field from the list.
 - c. The new field name will appear indented until you click **Apply** to save your changes.
3. Delete a sort field: Click the Delete button  next to any field to delete it from the list.
4. Define advanced sort options.

Important: We suggest that you not change advanced sort performance options without consulting your system administrator. If you change these options, you may negatively affect sort performance.

- a. Click **Advanced**.
- b. Use the **Override sort performance options** toggle to change your sort memory and temporary file settings.

- c. Use the **In memory record limit** field to define the caching and memory to keep before writing to disk.
- d. Define the **Maximum number of temporary files** to use for sorting.
- e. Use the **Compression** toggle to change your option for compressing files stored on disk.
- f. Click **OK** to save your changes.

You are ready to [Configure the Sort tab details - Read from File](#) on page 55.

Configure the Runtime tab details - Read from File

Use the Runtime tab to define record processing options for your flow.

1. Set the first record to process.
 - Use the **Starting Record** up and down arrows to select a record other than record 1 as the first record to process. You can use the arrows to set the starting record back to 1.
 - Alternatively, you can click the **Starting Record** field and enter the starting record number.
2. Define the number of records to process.
 - Select **All** records to process all input records (this is the default).
 - Select **Max Records** and define a maximum number of input records to process. Use the arrows to select a number or type a value in the entry field.
3. Click **Apply** to save your definitions.

you can now configure a Sorter stage if your flow is using one.

Related tasks

[Configure the Sorter stage](#) on page 77

These configuration steps apply to the Technical Preview version of Flow Designer. We will update the documentation for this stage when the full version of Flow Designer is available.

Configure the Input stage

Follow these steps to configure the Input stage.

1. Open an existing flow or start with a blank workspace.
2. Select an Input stage from the palette and drag the stage to the canvas.
3. Double-click the input stage to open the configuration view. Note that the input configuration has three tabs: **File**, **Sort**, and **Runtime**.
4. Select an input file for this stage.

You are ready to [Configure the File tab details - Read from File](#) on page 54.

Configure the File tab details - Input stage

1. At the **File Name** field, click the file name button to display the **Choose File** selection dialog. Flow Designer displays a list of available files at the default location.
2. To select an input file, you can:

- Go to another location using the navigation character (<).
- Filter the list of files using a letter or string then clicking the filter button.
- Click a to select a file name in the scroll box, and click **OK** to confirm your choice.

Flow Designer returns to the Read From File configuration page File tab, displaying your input file in the File **Name** field.

3. Review the populated File tab fields to ensure correctness and make adjustments if needed.
4. Check the boxes that provide more detail on the file content:
 - First row is header record – Toggle to **YES** if the first row in the input file is a header record.
 - Treat records with fewer fields than defined as malformed – Toggle to **YES** to implement this global record handling option. You can learn more about handling malformed records here: [Managing malformed input records](#) on page 25
5. Use the **File Schema** preview workspace to:
 - a) Change the order of the input fields:
 1. Click any gray bar in front of a field name; the pointer changes to a hand.
 2. Hold down the mouse button and drag the field selection to its new position.
 3. Release the mouse button when the new position is correct.
 4. Click **Regenerate** to save the change.
 - b) Rename a field – Double-click a field name to open a text input box, allowing you to change the name.
 - c) Change the field content type – Double-click the **Type** column for any field to display the drop-down, and select a valid format for the field content.
 - d) Selectively trim fields to their used length.
 - e) Selectively define a field's locale or format – If available, double-click the **Locale/Format** column for any field, and change the definition.
 - f) Selectively delete fields – Click the trash can: You will see a confirmation message before you can proceed.

Tip: Use the **Regenerate** button to re-examine and re-display the file schema after you make changes.

6. Click **Apply** so that your changes take effect, or click **Cancel** to return to the workflow canvas without saving changes.

You are ready to [Configure the Sort tab details - Input stage](#) on page 57.


Configure the Sort tab details - Input stage

The default view for the Sort tab assumes that your flow's sort order will copy the order in which those fields appear in the input file. You can change the defaults on this page.

Tip: You can click **Cancel** to abandon changes at any time.

1. Change the order of the sort fields.

For more information, see [Configure the File tab details - Read from File](#) on page 54.

2. Change the sort fields used.
 - a. Double-click the name of any sort field. Alternatively, click the field name once to select it and click the drop-down arrow to show a list of available sort fields.
 - b. Select a new field from the list.
 - c. The new field name will appear indented until you click **Apply** to save your changes.
3. Delete a sort field: Click the Delete button  next to any field to delete it from the list.
4. Define advanced sort options.

Important: We suggest that you not change advanced sort performance options without consulting your system administrator. If you change these options, you may negatively affect sort performance.

- a. Click **Advanced**.
- b. Use the **Override sort performance options** toggle to change your sort memory and temporary file settings.
- c. Use the **In memory record limit** field to define the caching and memory to keep before writing to disk.
- d. Define the **Maximum number of temporary files** to use for sorting.
- e. Use the **Compression** toggle to change your option for compressing files stored on disk.
- f. Click **OK** to save your changes.

You are ready to configure the **Runtime tab** on page 25 details.

Configure the Runtime tab details - Input stage

1. Set the first record to process.
 - Use the **Starting Record** up and down arrows to select a record other than record 1 as the first record to process. You can use the arrows to set the starting record back to 1.
 - Alternatively, you can click the **Starting Record** field and enter the starting record number.
2. Define the number of records to process.
 - Select **All** records to process all input records (this is the default).
 - Select **Max Records** and define a maximum number of input records to process. Use the arrows to select a number or type a value in the entry field.
3. Click **Apply** to save your definitions.

You can now configure a Sorter stage if your flow is using one.

Related tasks

Configure the Sorter stage on page 77

These configuration steps apply to the Technical Preview version of Flow Designer. We will update the documentation for this stage when the full version of Flow Designer is available.

Configure the Read from XML Stage for Subflows

Important: We suggest that you not change advanced sort performance options without consulting your system administrator. If you change these options, you may negatively affect sort performance.

1. Open an existing flow or start with a blank workspace.
2. Select a **Read from XML** stage from the palette and drag the stage to the canvas.
3. Double-click the stage to open the configuration view.
4. Use the **Override system-wide default options** toggle so that options defined for this service will not affect any other services.
5. Click **OK** to save your changes.

Configuring Control stages

Creating and configuring data transformations

Flow Designer functions provide the building blocks that help you to process, change, and convert data for site specific requirements. These custom transformations are configurable through the Transformer stage.

Note: This section provides a preview of the transforms available in Flow Designer. Note that only *custom* transforms are supported in this release of Flow Designer. Additionally, this preview version does not support multiple custom transforms. You can create Transforms in Enterprise Designer to use in this version of Flow Designer. For more information, see [Creating a Custom Transform](#) on page 34 in the *Dataflow Designer Guide*.

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 scripting. For more information about Apache Groovy, see groovy-lang.org.

Related concepts

[Transformer Options page](#) on page 60

Use the Transformer stage to perform common data transformations.

[Transformer stage - Transform reference](#)

Related tasks

[Creating a Custom Transform](#) on page 62

The Transformer stage has predefined transforms that perform multiple common data transformations. If the predefined transforms do not meet your needs, you can write a custom transform script using Groovy.

Transformer Options page

Use the Transformer stage to perform common data transformations.

When you add a Transformer stage to your flow and double-click the stage icon, you will see the Transformer Options page. The Transformer Options page is a workspace that allows you to construct transforms from a variety of functions that come with Flow Designer. This page consists of a palette on the left side, a scripting workspace, and **Apply** and **Cancel** buttons.

Table 3: Fields and settings on the Transformer Options page

Field or setting	Description	Options
Field and function display	This drop-down, at the top left of the page, displays all field and function options, or filters the display based on your selection.	Select the fields and functions to display: <ul style="list-style-type: none"> • All (no filtering) - Display all fields and functions • Input - Display all available input fields for the current input file • Output - Display all Output fields resulting from the current flow • Function - Display all available processing functions based on flow input and output
Filter entry field	For the selection in the Field and function display, apply a filter to narrow the fields or functions displayed. Click X in this field to clear the filtering and display all items in the category you selected in the Field and function display.	Enter a character or string to use for filtering the display. For example, if you chose Input in the Field and function display, then type "addr" in the filter entry field, the display might show "AddressLine1" and "AddressLine2".
Input fields	Use the arrow to expand and collapse the list of input fields available to process using a custom transform.	<ul style="list-style-type: none"> • Name - Field name • Type - Field data type • List - Select a field to use in a List transform.

Output fields	<p>Use the arrow to expand and collapse the list of output fields to process using a custom transform.</p> <p>To add a field:</p> <ol style="list-style-type: none"> 1. Click NEW to display a new row and input field. 2. Add the new field name; for example, "OutputAddr1". 3. Click the content of the Type field to apply the field format. 4. Check the List selection box to include this field in output transform processing. 	<ul style="list-style-type: none"> • Name - Field name • Type - Field data type • List - Select a field to use in a List transform. <p>You can click the trash can next to any output field to delete it.</p>
Functions	<p>Use the arrow to expand and collapse the list of functions. You can expand any function parent category to see the functions available under it.</p> <p>Hover over any function in an expanded category to see a description of the function, a list of applicable parameters, and a description of the data returned.</p> <p>For any selected field, double-click a function to add it to the scripting workspace.</p>	<p>The function categories are:</p> <ul style="list-style-type: none"> • DateTime - Processes data to derive a DATETIME object using INT64 values representing the year, month, day, hour, minute, and second. • Math - Performs mathematical/trigonometric calculations. • String - Performs processing on a defined data string producing a non-numerical result. • Miscellaneous - Includes random, roundTo, truncate, and truncateTo functions.
Scripting workspace	<p>This is the area where you will construct your data transformation scenarios.</p>	<p>For any selected field, double-click a function to add it to the scripting workspace.</p>
Apply and Cancel buttons	<p>Save or discard your data transformation.</p>	<ul style="list-style-type: none"> • Apply - Apply the function/transformation definition to the selected field. • Cancel - Discard the transformation scenario for the selected field.

Related tasks

[Creating a Custom Transform](#) on page 62

The Transformer stage has predefined transforms that perform multiple common data transformations. If the predefined transforms do not meet your needs, you can write a custom transform script using Groovy.

Creating a Custom Transform

The Transformer stage has predefined transforms that perform multiple 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.

Note: Review this topic to learn how to select fields for processing: [Creating a Custom Transform](#).

1. In Flow Designer, add a Transformer stage to the dataflow.
2. Double-click the Transformer stage to display the Transformer Options page.
3. Using the palette on the left side of the page:
 - a) Select one or more **Input fields** to which you will apply the transform.
 - b) 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 **NEW** in the field list.
4. Click in the **Script Editor** workspace.

The editor provides an easy-to-use workspace to develop your custom transforms, which includes type-ahead and selections. As you type in the workspace, Flow Designer provides applicable parameters based on your entries.

5. Use this reference as a guideline for constructing your custom transformation scenarios.

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="391 394 1459 478">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="391 594 1459 657">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="391 772 1459 804">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="391 1146 1459 1241">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="391 1356 1459 1472">int recordNumber = 1; data['Record_Number']= recordNumber; recordNumber++;</pre> <p>To change the scope of this variable to all input records, specify this:</p> <pre data-bbox="391 1545 1459 1692">import groovy.transform.Field @Field int recordNumber = 1; data['Record_Number']= recordNumber; recordNumber++;</pre>

Task	Instructions
<p>To create a new field using a numeric data type</p>	<p>Enter this code in the script editor:</p> <pre data-bbox="391 384 1459 447">data['FieldName'] = new constructor;</pre> <p>Where <i>constructor</i> is one of these:</p> <p>java.lang.Double(<i>number</i>) Creates a field with a data type of Double.</p> <p>java.lang.Float(<i>number</i>) Creates a field with a data type of Float.</p> <p>java.lang.Integer(<i>number</i>) Creates a field with a data type of Integer. You can also create a new integer field by specifying a whole number. For example, this will create an integer field with a value of 23:</p> <pre data-bbox="578 884 1403 947">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="391 1157 1459 1220">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="391 1354 1459 1417">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="578 1627 1403 1690">2013,12,23</pre> <p>com.pb.spectrum.api.datetime.Time(<i>hour,minute,second</i>)</p>

Task	Instructions
	<p>Creates a field with a data type of time. For example, 4:15 PM would be:</p> <pre data-bbox="578 401 1403 459">16,15,0</pre> <p>.</p> <p>com.pb.spectrum.api.datetime.DateTime(year,month,day,hour,minute,second)</p> <p>Creates a field with a data type of DateTime. For example, 4:15 PM on December 23, 2013 would be:</p> <pre data-bbox="578 663 1403 722">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="391 837 1195 905">data['TransactionDate'] = new com.pb.spectrum.api.datetime.Date(2013,12,23);</pre>
<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="391 1052 1459 1110">data['FieldName'] = true or false;</pre> <p>For example, to create a field named IsValidated and set it to false, you would specify this:</p> <pre data-bbox="391 1220 1459 1278">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="391 1461 1459 1860">NewListField = [] NewRecord = factory.create() NewRecord['NewField1'] = "Value" NewRecord['NewField12'] = "Value" ... NewListField << NewRecord NewRecord = factory.create() NewRecord['NewField1'] = "Value" NewRecord['NewField12'] = "Value" ...</pre>

Task	Instructions
	<pre data-bbox="394 310 1455 401">NewListField << NewRecord data['ListOfRecords'] = NewListField</pre> <p data-bbox="394 422 1455 485">For example, this creates a new list field called "addresses" consisting of two "address" records.</p> <pre data-bbox="394 506 1455 877">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> <p data-bbox="394 898 1455 1003">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="394 1024 1455 1178">phoneNumbers = [] phoneNumbers << data['HomePhone'] phoneNumbers << data['WorkPhone'] data['PhoneNumbers'] = phoneNumbers</pre>
<p data-bbox="224 1262 378 1367">To concatenate fields</p>	<p data-bbox="394 1262 1455 1335">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="394 1356 1455 1446">String fullname = data['FirstName'] + ' ' + data['LastName']; data['FullName']=fullname;</pre> <p data-bbox="394 1467 1455 1541">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="394 1562 1455 1682">address1 = data['AddressLine1']; address2 = data['AddressLine2']; data['Address']=address1+ ', ' + address2;</pre>
<p data-bbox="224 1766 378 1839">To parse a field</p>	<p data-bbox="394 1766 1455 1839">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</p>

Task	Instructions
	<p>five-character ZIP Code and the +4 portion and writes them to separate fields in the output record.</p> <pre data-bbox="391 401 1455 768"> 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>
<p>To perform conditional processing</p>	<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 data-bbox="391 1031 1455 1188"> city = data['City']; address1 = data['AddressLine1'] if(city.equals('Austin')) data['AddressCity']=address1 + ', ' + city; </pre>
<p>To perform looping</p>	<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>
<p>To augment data</p>	<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 data-bbox="391 1524 1455 1682"> firmname = data['FirmName']; constant = 'Incorporated'; if(firmname.length() > 0) data['FirmName']=firmname + ' ' + constant; </pre>

Task	Instructions
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>

6. After you are done entering your script, click **Apply** to save your work.

Related concepts

[Transformer Options page](#) on page 60

Use the Transformer stage to perform common data transformations.

Related reference

[Flow Designer stages and transforms](#)

Channel Mapping


This section describes the preview version of the channel mapping feature.

Channel mapping is the process of designating which data to move from an input source field into an output target field. With Flow Designer, you can:

- Map fields
- View field mapping details
- Unmap fields

Flow Designer indicates map and selection status in these ways:

Table 4: Channel and map status

Description	What it shows
If a channel contains no mapping, it appears as a solid gray line if it is not selected.	

If a channel is selected, it changes to a solid blue line.



If a channel contains mapping, but is not selected, it appears as a gray line containing a dot to show that mapping applies to that channel.



If a selected channel contains mapping, it appears as a solid blue line with a dot.



Hover on any channel to see the port names on each endpoint.



Mapping details page

Flow Designer displays mapping controls and status on the Mapping details page.

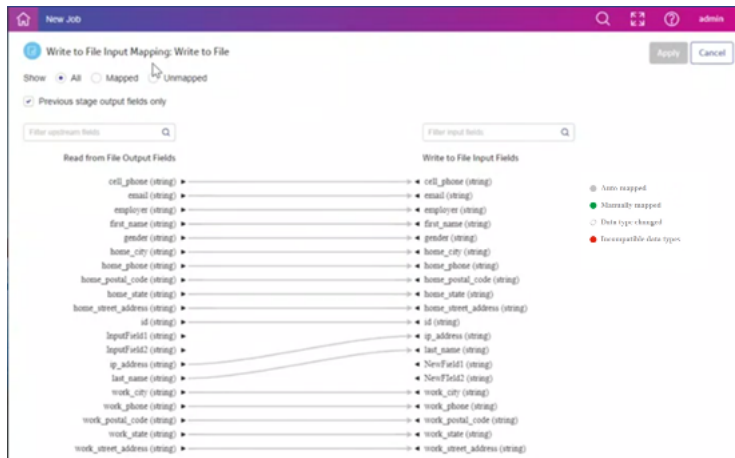







Table 5: Contents of the Mapping Detail page

Option	Description
Show	Filters the fields displayed: <ul style="list-style-type: none"> • All (default) • Mapped only • Unmapped only
Previous stage output fields only	Controls whether the output field list contains only fields from the previous stage (default) or all upstream fields
Filter input field names	Depending on the display you selected, shows a filter field that displays names based on a string you enter here
<i>Stage name</i> Output fields	Lists the output fields upstream of this channel
<i>Stage name</i> Input fields	Lists the input fields of the next stage in the dataflow
Mapping type	The mapping type is indicated by color or dashes. The legend on the page identifies channel colors and patterns.
 Auto-mapped	Auto-mapped Fields that Flow Designer mapped between output and input. These are represented by a grey line.
 Manually mapped	Manually mapped Fields that users mapped for a flow. These are represented by a green line.
 Data type changed	Data type changed Shows mapped fields that require data conversion between output and input. These are represented by a dashed line.
 Incompatible data types	Incompatible data types Fields that have incompatible data types. Incompatible data types cause errors during dataflow execution. These are represented by a red line.

View mapping details

Flow Designer displays mapping controls and status on the Mapping details page.

- Double-click the mapping indicator (dot) in a channel connection  to display the Mapping details page.

Manually map fields

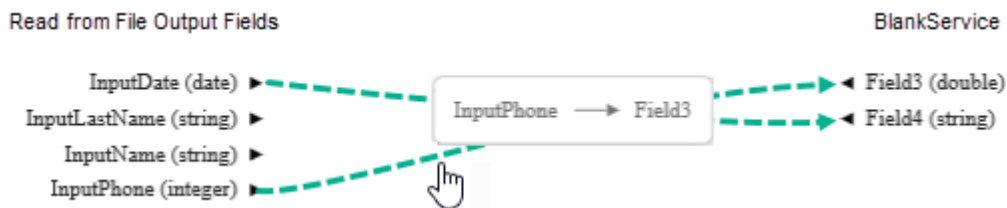
Flow Designer auto-maps fields, by default, based on its investigation of applicable fields.

You have the option to manually map fields to customize your output format and content.

1. With the Input Mapping page open, left-click the field name to map on the left/source field side.
2. Hold down the left mouse button and a line appears. Drag this channel (line) to the right-side target field to connect.
3. If the data types are compatible, Flow Designer displays a green line to show the manual field map you defined.



Data type conversions are represented by dashes. You can hover over any successfully mapping channel to display the mapping in a tooltip.



If the data types are incompatible, the channel is red.



4. If the channel is red, You must unmap or remap to a compatible field.

Note: For more information data type conversions, see [Defining Data Type conversion for mapped fields](#) on page 72.

5. Once you have correctly mapped your fields, click **Apply** to save your changes.

Change (re-map) fields

To re-map an auto-mapped field:

1. Click the channel between two auto-mapped fields. Flow Designer changes the channel to blue and adds a diamond handle on the channel, showing that the field can be remapped.
2. Drag the arrow to the new field, and release the mouse.

Unmap fields

Click once on any map channel (connector line), and press Delete.

Defining Data Type conversion for mapped fields

Spectrum Technology Platform automatically changes field data types as needed using the type conversion settings specified in Management Console, or through the dataflow type conversion options specified in Flow 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 a limited number of type conversions that you can perform manually:

- 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 data type, follow this procedure.

1. In Flow 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.
This displays field mappings between output fields in the upstream stage and input fields in the downstream stage. A dashed line indicates a data type change.
2. Double-click the mapping that you want to configure.
If type conversion options are configurable for the mapped fields, this expands the **Type Conversion Options** panel.
Note: The panel displays the appropriate options depending on the data type.
3. Click to toggle **Override type conversion options** to **ON**.
4. 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 for the date, time, and datetime field types. Your selection will also determine the language used when a month or day is spelled out. For example, if you specify English the first month of the year would be "January" but if you specify Spanish it would be "Enero".
5. In the **Format** field, specify the format you want to use for date, time, date and time, or numeric data that is converted to a string.
When the data, time, date and time, or numeric data is converted to a string, the string value will be in the format you specify here.

Field type	Do this
Date	<p>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.</p> <p>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 <code>3/2/20</code>.</p>
Time	<p>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.</p> <p>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 <code>11:00 PM</code>.</p>
DateTime	<p>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.</p> <p>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 <code>3/2/20 11:00 PM</code>.</p>
Whole numbers	<p>Select the formatting you want to use for whole numbers (data-types float and double).</p> <p>For example, if you choose the format #,### then the number 4324 would be formatted as <code>4,324</code>.</p> <p>If you leave this field blank, 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</p>
Decimal numbers	<p>Select the formatting you want to use for numbers that contain a decimal value (data-types integer and long).</p> <p>For example, if you choose the format ###0.0# then the number 4324.25 would be formatted as <code>4,324.25</code>.</p> <p>If you leave this field blank, 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</p>

Field type	Do this
	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 171. To specify your own number format, type the format into the file using the notation described in [Number Patterns](#).

6. Click **OK**.

The color of the data type name changes to green.

7. Click **Apply**, then the Save button  on the toolbar to save the change.

Configuring Record Combiner stage

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 the **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.

Option	Description
Left input stage	<p>From the dropdown, select the stage, records of which you want to use as the left table in the <code>JOIN</code> operation. The selected stage is displayed as Select left input field(s) for the join condition.</p> <p>All other Source stages move under Select right input field(s) that match <stage selected in the left table> field(s) for the join condition. Records from these stages 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>
Select left input field(s) for the join condition	Select the fields from the "Left" table for the <code>JOIN</code> operation
Select right input field(s) that match <stage selected in the left table> field(s) for the join condition	<p>Select the fields for the Right tables in the <code>JOIN</code> operation.</p> <p>Note: The valid data types for join fields are integer, string, datetime, date, long, float, double, and big decimal.</p>
Join type	<p>Select the type of <code>JOIN</code> operation you want to perform.</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> <p>Left Returns all records from the left port even if there are no matches between the left port and the other ports.</p> <p>Full Returns all records from all ports.</p>

Option	Description
Data from the previous stage is sorted	<p>Specifies whether the records are already sorted by the specified 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 specified 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. Select this option only when you are sure that the records in the left port are already sorted.</p>

11. Click the **Field Resolution** tab, and configure it.

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 is to be used 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 the **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.

Configure the Sorter stage

These configuration steps apply to the Technical Preview version of Flow Designer. We will update the documentation for this stage when the full version of Flow Designer is available.

You must configure an input file before you can apply a sorter stage.

1. Configure the input stages for your flow.
2. Locate the **Sorter** stage in the Stage Palette under Control Stages.
3. Click the **Sorter** stage and drag it to the canvas.
4. Connect the **Input** file and **Sorter** stages with a channel.
5. Double-click the **Sorter** stage to display the Sorter Options configuration page.
6. If not sort fields are listed, click Select to add a sort field to the sort options list.

Change order of sort fields

1. Change the order of the sort fields.
 - a. Click any gray bar in front of a field name; the pointer changes to a hand.
 - b. Hold down the mouse button and drag the field selection to its new position.
 - c. Release the mouse button when the new position is correct.
 - d. Click **Regenerate** to save the change.
2. Change the sort fields used.
 - a. Double-click the name of any sort field. Alternatively, click the field name once to select it and click the drop-down arrow to show a list of available sort fields.
 - b. Select a new field from the list.
 - c. The new field name will appear indented until you click **Apply** to save your changes.
3. Change the sort order for a field.
 - a) Double-click the sort direction for any sort field. Alternatively, click the Direction definition once to select it and click the drop-down arrow.
 - b) Select an option from the list.
 - c) The selection appears indented until you click **Apply** to save your changes.
4. Add and configure a sort field.
 - a) Click **Select**, which appears below the list of sort fields. You will see a list of available fields.
 - b) Select a field to add to the list.
 - c) Configure the sort direction: Ascending or Descending order.

Note: To learn more about how Flow Designer sorts data, review [How Flow Designer sorts data](#) on page 31
 - d) Click the **Sort Type** column to select the field's data type. This column is disabled if incoming data is not in string format (see [Data Types](#) on page 7 for more information).
 - e) To remove blank space from before and after the value before sorting, check the box in the Trim column.

Note: The trim option does not modify the value of the field. It only trims the value for sorting. Note that if your incoming data is not in string format, the Trim column is disabled.
 - f) Use the **Treat Null As** column define the placement of null values in the sorted list: largest or smallest. Placement depends on the combination of options selected in the **Order** and **Treat Null As** fields:

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

5. Repeat the above steps to add sort fields.
6. See Step 1 to rearrange the sort order. change the order to sort first by one field, then sort the resulting order using another field.
7. Click **Apply** to save your changes.
8. To delete a sort field, click the trash can button next to any field to delete it from the list.
9. Define advanced sort options.

Important: We suggest that you not change advanced sort performance options without consulting your system administrator. If you change these options, you may negatively affect sort performance.

- a. Click **Advanced**.
- b. Use the **Override sort performance options** toggle to change your sort memory and temporary file settings.
- c. Use the **In memory record limit** field to define the caching and memory to keep before writing to disk.
- d. Define the **Maximum number of temporary files** to use for sorting.
- e. Use the **Compression** toggle to change your option for compressing files stored on disk.
- f. Click **OK** to save your changes.

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.

To use the **Unique ID** stage, 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. In the dialog box that appears, give a name to the **Flow**, **Job**, **Service**, or **Subflow** you are creating.
4. From the **Palette Panel**, drag the **Unique ID** stage to the canvas.
5. Connect the required **Sources** (such as **Read from File**) and **Sink** (such as **Write to File**) stages.
6. Configure the **Sources** stage. See the documentation of the respective stage for field-level details.
7. Click **Unique ID**. The **Unique ID Generator Options: Unique ID Generator** page is displayed.

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. On the **Unique ID Generator Options: Unique ID Generator** page, modify the **Output field name**, if required. The default is **RecordID**.
2. By default **Sequential Numeric Tag; Starting at 0** is displayed as the method to generate the Unique ID. Click the corresponding **Modify**  icon to modify the method you want to use to generate the **Unique ID**.

Options	Description
Sequential Numeric tag starting at	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.
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.

3. To create a new rule, click the **Add Rule** button and specify the **Unique ID Field Options**, as described below.
 - a. **Algorithm:** Specifies one of these algorithms 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.

- 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 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.
- Soundex** Returns a Soundex code of selected fields. Soundex produces a fixed-length code based on the English pronunciation of a word.
- SpanishMetaphone** Returns a Metaphone coded key of selected fields for the Spanish language. This metaphone algorithm codes words using their Spanish pronunciation.
- Substring** Returns a specified portion of the selected field.
- b. **Field name:** 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
 - c. **Start Position Length:** These two field get enabled if you selected the Substring algorithm, and it allows you to specify the portion of the field you want to use in the substring.
 - In the **Start Position** field, specify the position in the field where you want the substring to begin.
 - In the **Length** field, select the number of characters from the start position that you want to include in the substring.

For example, say the data in the **LastName** field has `Augustine`. If you specified 3 as the start position and 6 as the length, the substring would produce: `gustin`.

d. **Pre-Processing options:**

1. Check the **Remove noise character** 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.
2. 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.

4. Click **Ok**.

The rule gets added below **Sequential Numeric Tag; Starting at 0** in the table.

5. Repeat the steps to define additional algorithms to produce a more complex ID.

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

6. Click **Apply**

A preview can be seen in the **Preview** section.

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.

To use algorithms, perform these steps On the **Unique ID Generator Options: Unique ID Generator** page:

1. Click the **Add Rule** button.
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. **Pre-Processing options:**
 - a. Check the **Remove noise character** 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.
 - b. 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.

- To modify the rule, click the corresponding **Modify** icon  and make the changes.

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.

- In the **Unique ID Generator Options: Unique ID Generator** page, on the **Rules** tab, click **Modify**.
- 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.

- Click **OK**.
- Click the **Add Rule** button.
- 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.

6. 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.
7. 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

8. 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.
9. **Pre-Processing options:**
 - a. Check the **Remove noise character** 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.
 - b. 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.
10. Click **OK** to save your settings.
11. 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.
12. To modify the rule, click the corresponding **Modify** icon  and make the changes.

Configuring output stages

Configuring the Output/Write to File stage

1. From the Stage palette, drag a Write to File (output) stage to the canvas.
2. Double-click the output stage to open the configuration view.

Configure the File tab to define the output file details.

Configure the File tab details - Write to File

Use the File tab to define details about your processing output.

1. At the **File Name** field, click the file name button to display the **Save File** selection dialog. Flow Designer displays a list of available files at the default location.
You can select an existing file to overwrite or define a new output file.
2. To define a new output file, enter the file name in the **File Name** field.

To select an existing file:

- a) Go to another location using the navigation character (<).
- b) Filter the list of files using a letter or string, then click the **Filter** button.
- c) Click to select a file name in the scroll box, and click **OK** to confirm your choice.

Flow Designer returns to the Write to File configuration page "File" tab, displaying your output file in the **File Name** field.

3. Review the populated File tab fields to ensure correctness and make adjustments if needed.
4. Select the options that provide more detail on the file content:
 - **First row is header record** – Check if the first row in the output file is a header record.
 - **Treat records with fewer fields than defined as malformed** – check to implement this global record handling option. You can learn more about handling malformed records here: [Managing malformed input records](#) on page 25.
5. Use the File Schema preview workspace to:
 - a) Change the order of the output fields:
 1. Click any gray bar in front of a field name; the pointer changes to a hand.
 2. Hold down the mouse button and drag the field selection to its new position.
 3. Click **Regenerate** to save the change.
 - b) **Rename a field** – Double-click a field name to open a text input box, allowing you to change the field's name.
 - c) **Change the field content type** (format) – Double-click the **Type** column for any field to display the drop-down, and select a valid format for the field content.
 - d) **Selectively trim fields to their used length** - Eliminate unused space in any field.

- e) **Add a field** – Click **Add** in the File schema view or click the **Quick Add** button above the table. Enter the new field name and click away from the row. You will see the new field name and can make adjustments to it.
- f) **Selectively delete fields** – Click the trash can: You will see a confirmation message before you can proceed.

Note: Click the **Regenerate** button to re-examine and re-display the file schema after you make changes.

- g) **Recalculate field position** – Select this option to recalculate the position and length of the fields when you modify, move, or remove a field in the output file. You can de-select this option if you do not want the position and length recalculated and instead want the fields to stay in their existing position after you edit the output file.

Note: This option is available only if your record type is **Fixed Width** or **Line Sequential**.

- 6. Click **Apply** so that your changes take effect, or click **Cancel** to return to the workflow canvas without saving changes.

You are ready to configure the Sort Tab details.


Configure the Sort tab details - Write to File

The default view for the Sort tab assumes that your flow's sort order will copy the order in which those fields appear in the output file. You can change the defaults on this page.

Tip: You can click **Cancel** to abandon changes at any time.

1. Change the order of the sort fields.
 - a. Click any gray bar in front of a field name; the pointer changes to a hand.
 - b. Hold down the mouse button and drag the field selection to its new position.
 - c. Release the mouse button when the new position is correct.
 - d. Click **Regenerate** to save the change.

Note: Flow Designer sorts data in ascending or descending order, according to the **ASCII sort order standard**.
2. Change the sort fields used.
 - a) Double-click the name of any sort field. Alternatively, click the field name once to select it and click the drop-down arrow to show a list of available sort fields.
 - b) Select a new field from the list.
 - c) The new field name will appear indented until you click **Apply** to save your changes.
3. Change the sort order for a field.
 - a) Double-click the sort direction for any sort field. Alternatively, click the **Direction** definition once to select it and click the drop-down arrow.
 - b) Select an option from the list.

- c) The selection appears indented until you click **Apply** to save your changes.
4. Delete a sort field: Click the Delete button  next to any field to delete it from the list.
5. Define advanced sort options.

Important: We suggest that you not change advanced sort performance options without consulting your system administrator. If you change these options, you may negatively affect sort performance.

- a) Click **Advanced**.
- b) Use the **Override sort performance options** toggle to change your sort memory and temporary file settings.
- c) Use the **In memory record limit** field to define the caching and memory to keep before writing to disk.
- d) Define the **Maximum number of temporary files** to use for sorting.
- e) Use the **Compression** toggle to change your option for compressing files stored on disk.
- f) Click **OK** to save your changes.

You are now ready to configure the Runtime tab options.


Configure the Runtime tab details - Write to File

Use the Runtime tab to define record processing options for your flow.

1. Set the first record to process.
 - Use the Starting Record up/down arrows to select a record other than record 1 as the first record to write. You can use the arrows to set the starting record back to 1.
 - Alternatively, you can click the **Starting Record** field and type the starting record number.
2. Define the number of records to process.
 - Select **All Records** to write all output records (this is the default).
 - Select **Max Records** and define a maximum number of output records to write. Use the arrows to select a number or type a value in the entry field.

Inspecting and testing flows

Checking a Flow for Errors

Flow 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 Flow Designer finds an error, it displays the **Validation** pane at the bottom of the 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 how input data is transformed and processed at any point in a flow.

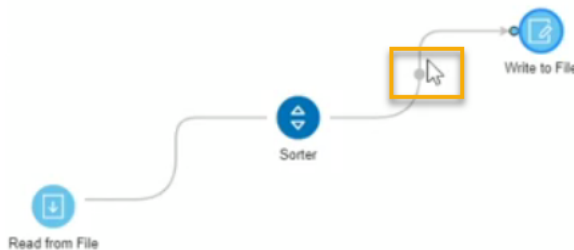
Note: For the preview version of Flow Designer, note these inspection limitations:

- This version supports single-point inspection. You cannot add multiple inspection points.
- Inspection is available for jobs, only. Flow Designer does not support for services.
- The input files must reside on the Spectrum server: Flow Designer does not support the inspection of local files at this time.

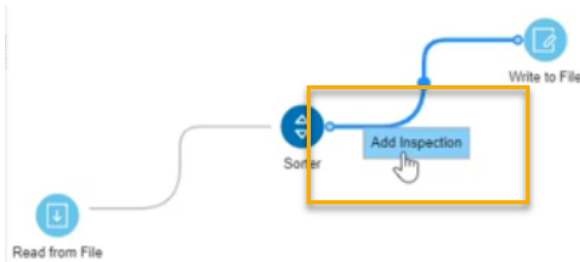
Inspection points provide a processing snapshot at any point in a flow. Inspection points allow you to confirm that the flow is producing the results you want, isolating problems, and identifying records that contain errors. The scenario below shows how to inspect data conversion of mapped fields during processing.

To add inspection points to your flow:

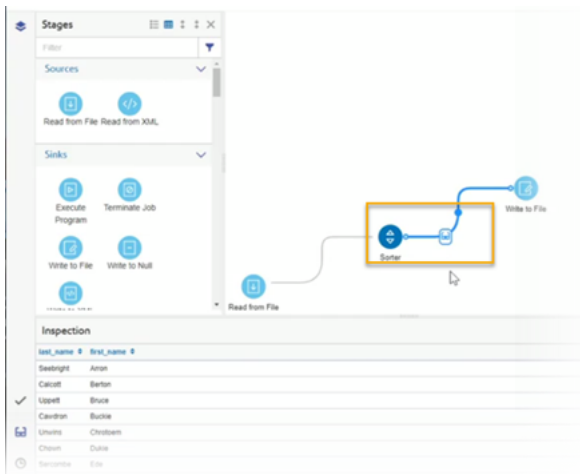
1. Open a flow containing an input file that resides on the server.
2. Perform a high-level check of the flow.
 - Are input and output fields mapped? Field mapping is indicated by a dot (node) on the channel between any two stages.




- You can hover over the mapping indicator to see the mapping that is taking place at that point in the flow.
 - Double-click the channel with the node, and it turns blue. Click that same channel once more to display the Mapping options page. Learn more about field mapping here: [View mapping details](#) on page 70.
 - Click at any section along the channel to see how the data will change for mapped fields during processing. Data changes include transforms, such as data type conversions. Flow Designer displays a hover message to provide that information.
3. To more closely inspect any point in the flow, right-click on a channel in the position where you will perform the inspection.



- Click again to insert the inspection point. Notice that this first inspection point occurs before the data conversion, to the left of the mapping indicator. Flow designer adds an eyeglass icon indicating the inspection point and displays the Inspection details panel at the bottom of the page.



The inspection panel includes controls  that refresh, collapse all, and expand all contents, and close the panel. If your input contains hierarchical data, you can extend the display to see parent-child data relationships. You can sort data by clicking any column. The sort order changes as you click on the column name.

When you add the inspection point, Flow Designer runs the job to produce a snapshot of that inspection point. You will see the data as it relates to that specific processing point. The preview provides a picture of how the data conversion takes place during the different phases of processing.

- To see the state of the data after mapping/conversion, place an inspection point on the other side of the mapping indicator.

Note: Changing the inspection point removes the previous inspection point. If you remove the inspection point, Flow Designer displays a message: There are no inspection points yet. Add an inspection point on a channel to view inspection data.



Once again, Flow Designer runs the job to take a snapshot of the data.

6. Observe the difference in field names this example, of the data before conversion and after conversion:

First inspection point, before processing

Inspection	
last_name	first_name
Seebright	Aaron
Calcott	Berton
Uppett	Bruce
Cawdron	Buckie
Unwins	Chrotoem
Chown	Dukie
Sercombe	Ede

Second inspection point, post-conversion

Inspection	
last	first
Seebright	Aaron
Calcott	Berton
Uppett	Bruce
Cawdron	Buckie
Unwins	Chrotoem
Chown	Dukie
Sercombe	Ede

Note: Right-click on any inspection point to delete it. You can also right-click on any point of the channel containing the inspection point to delete it. When you delete an inspection point, Flow Designer closes the inspection details panel.

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**.

Running flows

Run a job or flow using the Run Jobs button

This release of Flow Designer supports only input and output files that reside on the Spectrum Technology Platform server. Local files are not supported at this time.

There are two ways to run a job or flow: using the Run Job button or from the Execution History view.


1. Open the flow (job) to run on the canvas.
2. Save the flow to capture recent changes.
3. Click the Run job button .
4. After you start the job, you can observe the progress and these details.
 - **ID** – Job run number for this flow/job
 - **User** – Name of the user who started the flow/job
 - **Start Time** and **End Time**
 - **Duration**
 - **Results** - Green indicates that the job ran without errors. Red indicates errors that you will need to correct.
 - **Status** - Indicates whether the job is actively running or completed

Run jobs or flows from the Execution History panel

This release of Flow Designer supports only input and output files that reside on the Spectrum Technology Platform server. Local files are not supported at this time.

Only the jobs containing input and output files on server are supported. The jobs containing client side input and output files are not supported in this release.

There are two ways to run a job or flow: using the Run Job button or from the Execution History view.

1. Open the flow (job) to run on the canvas.
2. Click the Execution History button  to display the history for the current flow.
 - If someone ran the flow previously, you will see details for those job runs.
 - If no one has run the flow, you will see this message: `There is no execution history for this flow. Select the Run flow button to run the flow.`
3. Click the text "Run flow" to start the job.

4. After you start the job, you can observe the progress and these details.
 - **ID** – Job run number for this flow/job
 - **User** – Name of the user who started the flow/job
 - **Start Time** and **End Time**
 - **Duration**
 - **Results** - Green indicates that the job ran without errors. Red indicates errors that you will need to correct.
 - **Status** - Indicates whether the job is actively running or completed

Deleting flows

This topic provides the steps for deleting flows and describes the confirmation messages returned by the delete function.

1. Select the flow to delete using the check box next to the flow name.

Note: You may select multiple flows to delete at once.

2. Use the **delete flow** button to delete the selected flows from the server.

Flow Designer displays a confirmation message:

Deleting will permanently remove these flows from the server. Are you sure you wish to delete the selected flows?

3. If the flow you want to delete is not bound to any other flows, you can safely click **Yes**.
Some conditions prevent you from deleting flows. In those cases, Flow Designer returns an informational message. The message provides you with information to resolve the condition that prevents you from deleting selected flows.

If you received this message...

These flows cannot be deleted until issues are resolved: *flowname* is in use.

This is what is causing the error...

Other flows are using the named flows, and you cannot delete them.

Use the **View Impact Analysis** link to see more information on the flows that use the selected flow.

These flows cannot be deleted until issues are resolved: *filename* is checked out by *user*.

The named user locked the flows.

Coordinate with the named user before deleting the selected flows.

If you received this message...

These flows cannot be deleted until issues are resolved: *flowname* does not exist.



This is what is causing the error...

Someone already deleted the named flows from the server.

Refresh your screen to see all available flows.

Exposing flows

Exposing a flow to make it available for other users.

To expose a flow, open an existing flow or create a new flow on the Editor page, select the Save options button  next to the Save button  , and click **Save and expose** . This will make the flow available for use in other flows.

Note: If you have more than one version of the flow as a subflow, the version that is used in the parent flow is the exposed version. When you make a change to a subflow, be sure to expose the most recent version so your changes take effect in the flows that use the subflow.

Importing flows

Importing flows - import button method

1. Click the import button to open Windows Explorer.
2. Go to the location that contains the * .df files to import.
3. Select one or more files to import; click **Open**. Select one or more files to import; click **Open**.

Note: You can drop any exported flow anywhere on the Flow Designer workspace to import it. When you import a flow, Flow Designer automatically opens the flow in the editor.

Importing flows - drag-drop method

1. Open Windows Explorer in a separate window.
2. Go to the location that contains the *.df files to import.
3. To import files, drag them to the Import bin on the **Explorer** page.

The Import bin is the area that is labeled:

```
To import your flows, drag and drop your .df files here.
```

Note: You can drop any exported flow anywhere on the Flow Editor workspace to import it. When you import a flow, Flow Designer automatically opens the flow in the Flow Editor.

Exporting flows

Export any saved selected flow.

Flow designer exports the selected flows to the default browser download folder according to browser and user settings. Some browsers will prompt you to define a location to save the exported file.

You have the option to:

- Export the last saved version for the selected flow, regardless of its exposure status.
- Export the last exposed version of the selected flow.

Related concepts

[Exposing flows](#) on page 95

Exposing a flow to make it available for other users.

Related tasks

[Exporting the exposed version of a flow](#) on page 97

You can export the current exposed version of a flow. This allows you to make adjustments to a flow without changing the status of a saved flow, regardless of whether the flow is exposed to other users. Follow these steps to export a flow.

[Exporting the latest saved version of a flow - Explorer page](#) on page 97

You can export the latest saved version of a flow, regardless of whether the flow is exposed to other users. Follow these steps to export the latest saved version of a flow.



[Exporting the latest saved version of a flow - Editor page](#) on page 97

You can export the latest saved version of a flow, regardless of whether the flow is exposed to other users. Follow these steps to export the latest saved version of a flow.

Exporting the latest saved version of a flow - Explorer page

You can export the latest saved version of a flow, regardless of whether the flow is exposed to other users. Follow these steps to export the latest saved version of a flow.

Before you export a flow, you must save or save and expose the flow.

1. To select a flow to export, check the box next to any saved flow listed on the **Explorer** page. You can select multiple flows.
2. Click the Export types arrow  next to the Export icon .
3. Select **Export latest saved**.

You will see this message:

`Successfully exported flowname.`

Exporting the latest saved version of a flow - Editor page

You can export the latest saved version of a flow, regardless of whether the flow is exposed to other users. Follow these steps to export the latest saved version of a flow.

Before you export a flow, you must save or save and expose the flow.

Click the Export icon .

You will see this message:

`Successfully exported flowname.`

Exporting the exposed version of a flow

You can export the current exposed version of a flow. This allows you to make adjustments to a flow without changing the status of a saved flow, regardless of whether the flow is exposed to other users. Follow these steps to export a flow.

Before you export a flow, you must save or save and expose the flow.

These steps take place on the **Explorer** page.

1. To select a flow to export, check the box next to any saved flow listed on the **Explorer** page. You can select multiple flows.

2. Click the Export type arrow  next to the Export icon .
3. Select **Export exposed version**.

You will see this message:

```
Successfully exported flowname.
```

Related concepts

[Exposing flows](#) on page 95

Exposing a flow to make it available for other users.

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.
Timeout	Specifies whether to cancel the job if the command does not respond within a given amount of time: <ul style="list-style-type: none"> No timeout Do not cancel the execution if the command fails to respond. Timeout in milliseconds Cancels the execution attempt if the command does not respond in the specified number of milliseconds.

Option	Description
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 <code>C:\Java\jre7</code>. 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>

Using the template browser to create flows and jobs

Flow Designer provides a template browser that provides options for creating new jobs, services, or subflows on a blank canvas or using predefined templates.

Flow templates

Flow templates illustrate ways to 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.

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

Depending on the purpose of each template, the job may come with sample data. Or, it may be a service with no sample data. You can use flows in their original state and run those that are delivered as jobs to see how they function. Alternatively, you can change the flows' input and output files or add services into your jobs, adding other input and output files.

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

Flow Designer provides a selection of templates, available when you open the [New Flow page](#) on page 15.

Related concepts

[Types of Flows](#) on page 20


A flow is a series of operations that takes data from some source, processes that data, then writes the output to a destination. The processing of the data can be take the form of simple sorting to more complex data quality and enrichment actions.

Related tasks

[Create a new job, service, or subflow from a template](#) on page 101

The template browser provides predefined templates for creating jobs, services and subflows.

Create a new job, service, or subflow on a blank canvas

1. From the **Explorer** page, click the **New** icon . Flow Designer displays the [New Flow page](#) on page 15 with template selections.
2. Use the **Job**, **Service**, or **Subflow** links at the top left side of the page to display selections. Depending upon the number of templates available, you may have to use the scroll bar on the right side of the display to see all templates.
3. Optionally, you can use the filter feature to find templates, by name, for the type of flow you selected.
Type a letter or a string, such as "b" or "blank."
4. Double-click the blank template (this is the first template displayed at the top of the selection space) to open the flow on the canvas, or click the blank template and click **OK**.
Flow Designer prompts you to specify a file name. The application provides a unique name suggestion. For example, if "New Job 1" is already used, Flow Designer suggests "New Job 2." If you try to use an existing flow name, you will see an error prompting you for a different name.
5. Once you decide on a name, click **OK**.
The flow editor saves the new flow on the server and displays a blank workflow canvas, with your job name displayed at the top left of the workspace.

You can open the **Explorer** page to see if your flow is displayed. You may have to refresh the view to see the new flow. The initial display may show that the flow is locked by the current user (you), and displays the user's name under the **Locked by** column.

Related concepts

[Types of Flows](#) on page 20

A flow is a series of operations that takes data from some source, processes that data, then writes the output to a destination. The processing of the data can be take the form of simple sorting to more complex data quality and enrichment actions.


[Flow templates](#) on page 99

[New Flow page](#) on page 15

The **New Flow** page is the starting point for building new flows. Flow Designer provides a template browser that allows you to select templates to use in creating new flows.

Create a new job, service, or subflow from a template

The template browser provides predefined templates for creating jobs, services and subflows.

1. On the **Explorer** page, click the **New** icon . Flow Designer displays the **New Flow page** on page 15 with template selections.
2. Use the **Job**, **Service**, or **Subflow** links at the top left side of the page to display available selections in each category.
Depending upon the number of templates available, you may have to use the scroll bar on the right side of the display to see all templates.
3. Optionally, you can use the filter feature to find templates, by name, for the type of flow you want to build.
Type a letter or a string, such as "p" or "parseaddress."
4. Select a template, and click **OK**.
Flow Designer prompts you to specify a name. The application provides a unique name suggestion. For example, if "ParseAddressFields" is already used, Flow Designer suggests "ParseAddressFields1." If you try to use an existing flow name, you will see an error prompting you for a different name.
5. Once you decide on a name, click **OK**.
The flow editor saves the new flow on the server and displays the selected flow on the canvas, with all of the stages it contains. You will see your flow name displayed at the top left of the workspace.

You can now modify the template job to suit your needs.

Related concepts

[Types of Flows](#) on page 20

A flow is a series of operations that takes data from some source, processes that data, then writes the output to a destination. The processing of the data can be take the form of simple sorting to more complex data quality and enrichment actions.

[Flow templates](#) on page 99

[New Flow page](#) on page 15

The **New Flow** page is the starting point for building new flows. Flow Designer provides a template browser that allows you to select templates to use in creating new flows.

4 - Stages Reference

In this section

- Flow Designer stages.....103
- Control.....103
- Sources.....112
- Sinks.....113
- Module Stages.....115




Flow Designer stages










This is a reference to the stages available with Flow Designer.



Category	Description
Control on page 103	Control stages move data along different paths in a flow, to split or group records, and to perform basic data transforms and mathematical operations.
Sources on page 112	A source stage is the first stage in a flow. It defines the input data to process.
Sinks on page 113	A sink is the last stage in a flow. It defines what to do with output from the flow. A sink stage can also perform other actions to end a flow, such as executing a program.
Module Stages on page 115	Module stages are available through the modules installed on the Spectrum Technology Platform. Installed Modules appear as sections in the Stages panel.

Control

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

Stage	Icon	Description	For more information
Aggregator on page 105		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.	Aggregator is not available in this release.

Stage	Icon	Description	For more information
Broadcaster on page 105		Splits a single stream of records into multiple streams, allowing you to send records to multiple stages for simultaneous processing.	
Conditional Router on page 105		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.	Conditional Router is not available in this release.
Group Statistics on page 106		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.	Group Statistics is not available in this release.
Math on page 106		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.	Math is not available in this release.
Record Combiner on page 106		Record Combiner is a Control stage that uses one or more input ports to merge multiple records from multiple streams into a single survivor record based on a commonality.	Record Combiner on page 106
Record Joiner on page 106		Record Joiner performs a SQL JOIN operation to combine records from different streams based on a relationship between fields in the streams.	Record Joiner on page 106
Sorter on page 106		Use the Sorter stage to sort records using fields you specify.	Configure the Sorter stage on page 77
Splitter on page 106		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.	Splitter is not available in this release.
Stream Combiner on page 106		Joins multiple record streams 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 Note: Not available in this release.

Stage	Icon	Description	For more information
Transformer on page 107		Flow Designer provides transformer building blocks in the form of functions that allow you to process, change, and convert data to suit your requirements.	Creating and configuring data transformations on page 59
Unique ID Generator on page 111		The Unique ID Generator stage creates a unique key that identifies a specific record.	Unique ID Generator on page 111

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.

Broadcaster

Splits a single stream of records into multiple streams, allowing you to send records to multiple stages for simultaneous processing.

Conditional Router

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`).

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.

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.

Record Combiner

Record Combiner is a Control stage that uses one or more input ports to merge multiple records from multiple streams into a single survivor record based on a commonality.

Record Joiner

Record Joiner performs a SQL `JOIN` operation to combine records from different streams based on a relationship between fields in the streams.

Sorter

Use the Sorter stage to sort records using fields you specify.

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.

Stream Combiner

Joins multiple record streams 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.

Transformer

Flow Designer provides transformer building blocks in the form of functions that allow you to process, change, and convert data to suit your requirements.

Transformations are configurable through the Transformer stage. This section provides a preview of the transforms available in Flow Designer. Note that only custom transforms are supported in this release of Flow Designer.

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 Transform2 . For users of Spectrum Spatial , custom transforms can access spatial datasets. For more information, see the Stages section in the <i>Spectrum Spatial Guide</i> Spatial Stages .
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 .
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>
      </club>
    </clubs>
  </sports>
</sports_details>
```

```

    <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>
```



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.

Sources

A source stage is the first stage in a flow. It defines the input data to process.

Stage	Icon	Description	For more information
Read from File on page 112		This stage specifies an input file for a job or subflow. This stage is not available for services.	Read from File on page 112
Read from XML on page 113		This source stage specifies an XML-format input file for a job or subflow. This stage is not available for services.	Read from XML on page 113

Read from File

This stage specifies an input file for a job or subflow.

Not available for services.

Note: If you want to use an XML file as input for your flow, use the [Read from XML](#) on page 113 stage instead of Read from File. If you want to use a variable format file as input, use the [Read from Variable Format File](#) on page 113.

Configure the Read from File Stage

Follow these steps to configure the Read from File stage.

1. Open an existing flow or start with a blank workspace.
2. Select an Input stage from the palette and drag the stage to the canvas.
3. Double-click the input stage to open the configuration view. Note that the input configuration has three tabs: **File**, **Sort**, and **Runtime**.
4. Select an input file for this stage.

You are ready to [Configure the File tab details - Read from File](#).

Read from XML

This source stage specifies an XML-format input file for a job or subflow.

Not available for services.

Configure the Read from XML Stage for Subflows

Important: We suggest that you not change advanced sort performance options without consulting your system administrator. If you change these options, you may negatively affect sort performance.

1. Open an existing flow or start with a blank workspace.
2. Select a **Read from XML** stage from the palette and drag the stage to the canvas.
3. Double-click the stage to open the configuration view.
4. Use the **Override system-wide default options** toggle so that options defined for this service will not affect any other services.
5. Click **OK** to save your changes.


Read from Variable Format File





This stage specifies a variable-format input file.

Not available for services

Sinks

A sink is the last stage in a flow. It defines what to do with output from the flow. A sink stage can also perform other actions to end a flow, such as executing a program.

Stage	Icon	Description	For more information
Execute Program on page 115		This stage runs an external application as part of a process flow.	Defining Service Output

Stage	Icon	Description	For more information
Terminate Job on page 115		This stage is used in combination with Conditional Router to end a job if certain criteria are found within a record.	Terminate Job on page 115
Write to File on page 115		This stage writes output of a flow to a specified file.	Write to File on page 115
Write to Null on page 115		This stage sends output to an XML-format output file that can be consumed by other processes or flows.	Write to Null on page 115
Write to XML on page 115		This stage writes output from a flow or subflow to an XML file.	Write to XML on page 115

Execute Program

This stage runs an external application as part of a process flow.

Terminate Job

This stage is used in combination with Conditional Router to end a job if certain criteria are found within a record.

Write to File

This stage writes output of a flow to a specified file.

Write to Null

This stage sends output to an XML-format output file that can be consumed by other processes or flows.

Write to XML

This stage writes output from a flow or subflow to an XML file.

Module Stages

Module stages are available through the modules installed on the Spectrum Technology Platform. Installed Modules appear as sections in the **Stages** panel.




Module	Description
Context Graph Stages on page 116	Context Graph stages are available if you have licensed the Context Graph. Context Graph stages provide operations to manipulate Context Graph models in workflows.





Module	Description
Data Stewardship stages on page 150	These stages are available with Data Stewardship.
Enterprise Data Integration on page 151	These stages are available to you if you have licensed for the Enterprise Data Integration module. Use these to connect to data in multiple sources either directly or through integration with your existing data access technologies.
Data Quality on page 153	These stages are available if you have licensed the respective module, such as Advanced Matching Module. Use these stages to ensure the accuracy, timeliness, completeness, and consistency of the data used by your organization.

Context Graph

Context Graph Stages

Context Graph stages are available if you have licensed the Context Graph. Context Graph stages provide operations to manipulate Context Graph models in workflows.

Stage	Icon	Description	For more information
Delete from Model		The Delete from Model stage deletes entities and relationships from a model.	<ul style="list-style-type: none"> • Delete from Model on page 117
Import to Model		The Import to Model stage populates data in a new or existing Context Graph model from two incoming channels.	<ul style="list-style-type: none"> • Import to Model
Merge Entities		The Merge Entities stage merges two or more entities into a single entity.	<ul style="list-style-type: none"> • Merge Entities

Stage	Icon	Description	For more information
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. It executes a saved query for a model in a workflow. The output can be returned as flat or hierarchical data representing query levels.	<ul style="list-style-type: none"> • Query Model on page 123
Read from Model		The Read From Model stage executes a query to read data from an existing model.	<ul style="list-style-type: none"> • Read from Model
Split Entity		The Split Entity stage splits one entity into two or more new entities.	<ul style="list-style-type: none"> • Split Entity
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.	<ul style="list-style-type: none"> • Write to Model on page 128

Delete from Model

The Delete from Model stage deletes entities and relationships from a 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 port contains data for the deleted entities and relationships. The other port contains data for the records that were not deleted.

To configure a Delete from Model stage, you first select the model you want to modify, then choose whether to delete entities, relationships, or both entities and relationships, and finally complete options for the selected operations.

Delete from Model ports

The Delete from Model stage ports support one input channel and two optional output channels.

Input

The Delete from Model stage requires that your dataflow include an input stage that contains data from or queries the model whose elements you are deleting. The input stage could be a Read from Model stage, a Query Model stage, a source stage of some kind, or a control stage that combines or processes channels from input stages.

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. The other sink collects any records that the dataflow did not process correctly. This is called the Error Port. Records that pass through this port into the sink are considered malformed.

Capturing malformed records can help you identify problems 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.

Delete from Model options

The following sections describe how to configure the Delete from Model stage settings to delete entities and relationships from a Context Graph model.

- [General options](#) on page 118
- [Delete entities options](#) on page 119
- [Delete relationships options](#) on page 120

General options

These Delete from Model settings specify the model in which to perform deletions; whether to delete entities, to delete relationships, or to delete both entities and relationships; and whether to allow other stages to write to the model while the Delete to Model stage executes.

Setting	Option	Description
Model		Choose the model from which to remove entities or relationships.
Delete		Choose whether to delete entities, relationships, or both entities and relationships from the selected model.

Setting	Option	Description
	Entities	Choose this option to delete entities from the model. After you choose this option, configure the Entities settings to specify the entities to be deleted from the model.
	Relationships	Choose this option to delete relationships from the model. After you choose this option, configure the Relationships settings to specify the relationships to be deleted from the model.
	Entities and Relationships	Choose this option to delete both entities and relationships from the model. After you choose this option, configure both the Entities and Relationships settings to specify entities and relationships to be deleted from the model.
Write mode		Choose whether multiple Context Graph stages may simultaneously write to the selected model.
	Concurrent writes	This option permits the model to be written to by other Context Graph stages while this stage is performing deletions. This is the default setting for this stage operation.
	Exclusive lock	This option prevents other Context Graph stages from writing to the model while this stage is performing deletions.

Delete entities options

These settings identify entities that the Delete from Model stage deletes from a Context Graph model. Choosing **Entities** or **Entities and relationships** in the **Delete** box expands the following settings.

Location	Setting	Option	Description
	Location of entity type data		The setting in this box specifies where the entity type is located in the input data.
		In the entity ID field	Choose this setting if the entity type and entity label are in the same input data field. This field is known as the entity ID field, such as <code>_stp_id</code> .

Location	Setting	Option	Description
		Entity ID field	Choose the input data field that contains both the entity type and entity label. This is often, but not always, the <code>_stp_id</code> field.
	In a separate field		Choose this setting if the entity type and entity label are in different input data fields.
		Entity type field	Choose the input data field that contains only the entity type. This is often, but not always, the <code>_stp_type</code> field.
		Entity label field	Choose the input data field that contains only the entity label. This is often, but not always, the <code>_stp_label</code> field.
	Literal		Choose this setting to select a specific entity type.
		Entity type	Choose the appropriate entity type.
		Entity label field	Choose the input data field that contains only the entity label. This is often, but not always, the <code>_stp_label</code> field.

Delete relationships options

The **Relationships** settings identify relationships to delete from a Context Graph model.

Choosing **Relationships** or **Entities and relationships** in the **Delete** box expands three categories of settings:

- **Source entity** on page 120 settings identify source entities of relationships to be deleted.
- **Relationships** on page 121 settings identify relationships to be deleted.
- **Target Entity** on page 122 settings identify target entities of relationships.

Relationships that match the criteria defined by all three categories are deleted from the model.

Source entity

These settings identify source entities for relationships in the model. Choosing one of the **Location of entity type data** options expands settings for the selected data location.

Location	Setting	Option	Description
	Location of entity type data		The setting in this box specifies where the source entity type is located in the input data.
	In the entity ID field		Choose this setting if the source entity type and source entity label are in the same input data field. This field is known as the source entity ID field, such as <code>source._stp_id</code> .
		Entity ID field	Choose the input data field that contains both the source entity type and source entity label. This is often, but not always, the <code>source._stp_id</code> field.
	In a separate field		Choose this setting if the source entity type and source entity label are in different input data fields.
		Entity type field	Choose the input data field that contains only the source entity type. This is often, but not always, the <code>source._stp_type</code> field.
		Entity label field	Choose the input data field that contains only the source entity label. This is often, but not always, the <code>source._stp_label</code> field.
	Literal		Choose this setting to select a specific entity type.
		Entity type	Choose the appropriate entity type.
		Entity label field	Choose the input data field that contains only the source entity label. This is often, but not always, the <code>source._stp_label</code> field.

Relationships

These settings identify relationships between the source and target entities.

Setting	Description
Relationship label field	Choose the field that contains the relationship label to be deleted. This is typically, but not always, <code>Step1.stp_label</code> .
Unique ID is in a separate field	Check this check box if there are multiple relationships with the same label between the source and target entities, and choose the field that contains the unique ID. This is typically, but not always, <code>Step1.stp_key</code> .

Target Entity

These settings identify target entities for relationships in the model. Choosing one of the **Location of entity type data** options expands settings for the selected data location.

Location	Setting	Option	Description
	Location of entity type data		The setting in this box specifies where the target entity type is located in the input data.
		In the entity ID field	Choose this setting if the target entity type and target entity label are in the same input data field. This field is known as the target entity ID field, such as <code>target._stp_id</code> .
		Entity ID field	Choose the input data field that contains both the target entity type and target entity label. This is often, but not always, the <code>target._stp_id</code> field.
		In a separate field	Choose this setting if the target entity type and target entity label are in different input data fields.
		Entity type field	Choose the input data field that contains only the target entity type. This is often, but not always, the <code>target._stp_type</code> field.
		Entity label field	Choose the input data field that contains only the target entity label. This is often, but not always, the <code>target._stp_label</code> field.

Location	Setting	Option	Description
	Literal		Choose this setting to select a specific entity type.
		Entity type	Choose the appropriate entity type.
		Entity label field	Choose the input data field that contains only the target entity label. This is often, but not always, the <code>target._stp_label</code> field.

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. It executes a saved query for a model in a workflow. The output can be returned as flat or hierarchical data representing query levels.

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 hub.


Note: The Query Model stage in Flow Designer only supports saved queries. If you open a job that was created in Enterprise Designer that uses an unsaved query, you will be prompted to save the query.

How to

Add the Query Model stage to a workflow

This procedure steps through addition of the Query Model stage to a workflow in Flow Designer. Use the Query Model stage to execute a saved Context Graph model query in a workflow.

1. In Flow Designer, on the **Stages** panel, expand the **Context Graph** section, and drag the **Query Model** stage to the canvas.
2. Connect a source to the input port on the **Context Graph** stage.
3. Double-click the **Query Model** stage.
4. Specify the Context Graph **Model** and **Query** that you want to execute in the workflow. For more information, see [Query Model options](#) on page 124.
5. On the **Input Fields** tab, you can view input fields for the query. For more information, see [Input Fields tab](#) on page 125.
6. Click the **Output Fields** tab to configure output from the stage. The **Query results** box lists results that correspond to levels in a saved query. For more information, see [Output Fields tab](#) on page 125.

7. In the **Query results** box, click to view the output fields corresponding to the result in the adjacent stage fields table.
You can check or uncheck boxes in the table to include or remove fields in the output. Press the Ctrl or Shift key and click another result to simultaneously view fields for more than one result.
8. To configure the settings for any result listed in **Query Results** box, check the **Override query results** check box, and click a single result to configure its settings.
9. In the **Result Configuration** box, you can change the result name and path, whether to use type names on the fields, and choose flat or hierarchical list output for each result.
For more about these options information, see [Result Configuration](#) on page 126.
10. When you finish configuring the stage, click the **Apply** button.
11. If you have not done so already, connect the output port to a downstream stage.
The Query Model stage has a single required output port that you must connect to a stage that accepts input, such as the Write to File or Write to DB stage for flat output or the Write to XML stage for hierarchical output.
12. Before exiting the workflow, be sure to click the Save button  to save the workflow.

Reference

Stage ports

The Query Model stage ports support one input channel and one output channel.

Input

The Query Model stage requires that the input channel of your dataflow has defined the input fields. Any input fields or output fields accessed using the data command need to be defined in the **Fields** tab in the input and output channels. Otherwise, they will not appear as input and output fields in other stages in your dataflow.

Output

The Query Model stage has a single outgoing port to which you must attach a stage that takes input. Output fields are specified on the **Output Fields** tab. Output may be either flat or hierarchical, depending on the **List** setting in the **Result Configuration** (for more information, see [Output Fields tab](#) on page 125).

Query Model options

These settings specify the model, a saved query, and the maximum number of records that will be returned by the stage. The **Input Fields** and **Output Fields** tabs define input and output fields for the stage.

Model Choose the Context Graph model that you want to query. The drop-down list shows models that are currently in Context Graph.

Query Choose a saved query. This option is available only after you select a model in the **Model** box. The Query Model settings are for saved queries created for a model using the visual interfaces on the Context Graph Canvas or in Relationship Analysis Client. Queries created in the Query Model stage in Enterprise Designer have limited functionality in Flow Designer.

Note: The Query Model stage in Flow Designer does not support unsaved queries. If you open a workflow that contains unsaved queries created in Enterprise Designer, you will be prompted to save the queries before you can edit the workflow in Flow Designer.

Limit Specifies that maximum number of items that will be returned from the model by the stage per input record. The stage truncates any further output after it returns the maximum number of items.

Related reference

Input Fields tab on page 125

The **Input Fields** settings show input fields used by the query.

Output Fields tab on page 125

The **Output Fields** settings define fields and field names in the output. These settings also define whether the stage output is in flat or hierarchical format.

Input Fields tab

The **Input Fields** settings show input fields used by the query.

Filter Enter any portion of a field name to filter input fields shown in the table.

Name This column shows name of each input field in the query.

Type This column shows the data type of each input field in a query.

Output Fields tab

The **Output Fields** settings define fields and field names in the output. These settings also define whether the stage output is in flat or hierarchical format.

Override query results Check this check box to configure the result configuration. The default is to use the query result names defined by the query as they initially appear in the **Query Results** list. The order of result names in the list defines the hierarchical output. Check this check box to change result names, edit paths, or create flat instead of hierarchical output. To write flat data output, you can use the Write to File or Write to DB stages. To write hierarchical output, you can use the Write to XML stage.

Note: Query results overrides are not supported by Enterprise Designer. If you configure query result overrides in Flow Designer, the overrides will not appear if you open the workflow in Enterprise Designer.


Query results This box lists query results associated with each step in a saved query. The default result names are typically `Root`, `Step1`, `Step2`, and so forth, although they may have other names depending on the query.

- Click to select a result and view how data is returned in output fields for the corresponding step in the query. If the **Override query results** check box is selected, you can click a result to edit **Result Configuration** options for that result.
- Select multiple results to view combined output fields for the corresponding steps in the query. Press Ctrl and click to add a result to the selection, or press Shift and click to add contiguous results to the selection.

Result Configuration



These options allow you to edit the name and path for the result. You can also choose to output field values in a hierarchical list or as flat data. You must check the **Override query results** check box to edit these options for a the result that is currently selected in the **Query Results** box.

Include in results	Check or uncheck this check box to include or remove a result from the output. Clearing this check box removes the corresponding result fields from the stage field table. Do not uncheck this check box for the last result in a series or when there is only one result in the series. This check box is selected by default.
Name	The name for a result in the output. The default is the result name initially shown in the Query Results box. If the List check box is checked, this name designates a level in the hierarchy. It is the name of a level in hierarchical output, and by default, it also prefixes the property name for each output field.
Path	Shows the path for a result in the output hierarchy. By default, the path incorporates names of the preceding results. For any result, you can click to select one of the paths from a preceding result and thereby change the output hierarchy. For example, where the first result is named <code>Locations</code> and the second result is <code>CountryName</code> , the Path box shows "/" when you select <code>Locations</code> in the Query Results box. Click <code>CountryName</code> in the Query Results box, and Path box now shows <code>/Locations</code> . If you then click the third result, the Path box would show <code>/Locations/CountryName</code> . If you click <code>CountryName</code> in the Query Results box, click the Path box, then select "/" on the drop-down menu, this moves the <code>CountryName</code> result up the hierarchy from <code>/Locations/CountryName</code> to <code>/CountryName</code> . You can view the altered hierarchy reflected in the stage fields table.
Use type names	Check this check box to include the entity or relationship type name for a step in the output. Entities use the entity type and relationships use the relationship label for the type name. For example, if you do not select this option and the result name that shows in the Name box is <code>Afghanistan</code> , then the output fields for <code>Latitude</code> and <code>Date</code> properties would be <code>Afghanistan.Latitude</code> and <code>Afghanistan.Date</code> . If you select this option, then the entity type name, <code>Person</code> for this example, will prefix the property name. The output fields would then be <code>Afghanistan.Person.Latitude</code> and <code>Afghanistan.Person.Date</code> . You can use this option to help identify hierarchical relationships in flat data output. You cannot simultaneously select both the Use type names and List check boxes.
List	Check this check box to create hierarchical output for the currently selected result. The result name then forms one level in the hierarchy and the output fields for the result are inserted as its children. Uncheck this check box to combine output fields for a result with the output fields of the previous result. Clear this check box for every result for an entirely flat output. In both flat and hierarchical outputs, result names prefix output field names

unless you choose to remove them by clicking **Remove result name** on the edit menu  above the stage fields table.

Stage fields table

This table shows the result hierarchy. It also shows output fields for any results that are currently selected in the **Query Results** box. In this table, you can select the check boxes to include results and fields in the output. You can edit individual field names. You also can remove the result name prefix from the field names. The hierarchy shown in the table depends on the **Result Configuration** settings for each result in the **Query Results** results list.

 Remove result name	To remove the result name prefix from output fields, click the edit menu  above the stage fields table, then click Remove result name . Result names are prefixed to output field names. They are especially useful to distinguish fields and avoid conflicts when hierarchical data is flattened and output to rows. You can use this option to remove the result names if they are not needed.
Include	Check this check box to include a field in the output. Uncheck a check box to exclude a field from the output. Uncheck or check the check box next to a result name to uncheck or check all fields under that result. This can speed configuration when you want check the check box for one or two fields amongst many fields.
Stage Field	Shows the name of a field included in the output. You can click any field to edit its name. By default, the name concatenates a result name and a property name separated by a period. For example: <code>Person.Associate</code> or <code>Person._stp_id</code> for an entity. The property name is prefixed by the type name followed by a period if the Use type name setting is selected for the query result. For example: <code>Person.Person.Associate</code> or <code>Person.Person._stp_type</code> (in this example, the first <code>Person</code> is the result name and the second <code>Person</code> is entity type). For a relationship, the type name is the relationship label.
Property	The entity or relationship property name from the model that corresponds to an output field.
Type	Specifies the type of output field. This may be any one of the data types for properties (boolean, string, double, float, integer, long, date, time, datetime, or bigdecimal).

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.

How to

Create a new model

This procedure steps through how to use input data to create a new Context Graph model in Flow Designer with the Write to Model stage.

Before you perform this procedure, configure the source and control stages in a data flow to define and process the input data that you want to write to the Context Graph.


1. In Flow Designer, drag the **Write to Model** stage icon from the **Context Graph** panel to the canvas.
2. Optional: Click the default "Write to Model" stage icon label to give the stage a more meaningful name.
3. Drag an upstream connection to the input port on the **Write to Model** stage icon.
4. Double-click the **Write to Model** icon to open its options.
5. In the **Model** box, type the name for the new model.
6. In the **Write mode** box, click to select **Initial load**.
7. Optional: If you want to remove entities without relationships from the new model before closing the stage, check the **Remove orphaned entities** check box.
8. Add and configure new entity types that you want to include in the model.
 - For the procedure to do this, see [Configure a new entity type](#) on page 130.
9. Add relationships to the model.
 - For the procedure to do this, see [Configure new relationships](#) on page 131.
10. Click the **Indexes** button to configure indexes for the configured elements.
For more information, see [Indexes](#) on page 134.
11. Click **Apply**.
12. On the toolbar, click the Save button .

Update an existing model

This procedure steps through how to use input data and existing metadata to update a model.

Before you perform this procedure, configure the source and control stages in a data flow to define and process the source data that you want to write to an existing Context Graph model.

1. In Flow Designer, drag the **Write to Model** stage icon from the **Context Graph** panel to the canvas.
2. Click the default "Write to Model" stage icon label to give the stage a more meaningful name.
3. Drag an upstream connection to the input port on the Write to Model stage icon.

4. Double-click the **Write to Model** icon to open its settings.
5. In the **Model** box, click to select an existing model from the drop-down list.
6. In the **Write mode** box, click Exclusive lock or Concurrent write mode.
For more information, see [Write mode](#) on page 134.
7. Optional: If you want to remove entities without relationships from the model before closing the stage, check the **Remove orphaned entities** check box.
8. Add and configure existing or new entity types that you want to include in the model.
 - To configure new entity types in a model, see [Configure a new entity type](#) on page 130.
 - To configure entity types already in a model, see [Configure an existing entity type](#) on page 131.
9. Configure relationships in the model.
 - To configure new relationships in a model, see [Configure new relationships](#) on page 131.
 - To configure relationships already in a model, see [Configure existing relationships](#) on page 132.
10. Click the **Indexes** button to configure indexes for the configured elements.
For more information, see [Indexes](#) on page 134.
11. Click **Apply**.
12. On the toolbar, click the Save button .

Use the model schema to update an existing model

You can use the schema of an existing model to add and configure entities and relationships for the Write to Model stage. The schema guides adding entities and relationships to the stage.

Before you perform this procedure, configure the source and control stages in a data flow to define and process the source data that you want to write to an existing Context Graph model.

1. In Flow Designer, drag the **Write to Model** stage icon from the **Context Graph** panel to the canvas.
2. Click the default "Write to Model" stage icon label to give the stage a more meaningful name.
3. Drag an upstream connection to the input port on the Write to Model stage icon.
4. Double-click the **Write to Model** icon to open its settings.
5. In the **Model** box, click to select an existing model from the drop-down list.
After you complete this step, the **Show schema** toggle switch appears in the top right. It is initially set to **NO**.
6. In the **Write mode** box, click Exclusive lock or Concurrent write mode.
For more information, see [Write mode](#) on page 134.
7. Optional: If you want to remove entities without relationships from the model before closing the stage, check the **Remove orphaned entities** check box.
8. Click the toggle to switch **Show schema** to **YES**.

This displays entities and relationships from the selected model on the canvas. Items appear dimmed until they are configured.


9. Click dimmed entities to configure and add them to the stage.
 - To configure entity types from model schema, see [Configure an existing entity type](#) on page 131.

Entities appear undimmed after they are configured.

10. Click dimmed relationships between configured entities to configure and add them to the stage.
 - To configure relationships from a model schema, see [Configure existing relationships](#) on page 132.

Before you can configure a connecting relationship, both the source and target entity must be configured.

Relationships appear undimmed after they are configured.

11. Click **Apply**.
12. On the toolbar, click the Save button .

Completing this procedure saves configured entities and relationships from the model schema in the Write to Model stage. Entities and relationships that have not been configured continue to appear dimmed on the canvas. Dimmed items are not saved in the Write to Model stage.

Configure a new entity type

This procedure describes how to configure a new entity type in the Write to Model stage.

1. In the **Entity Types** box, drag the **New Entity Type** icon to the canvas. Completing this step expands the **Entity Configuration** panel.
2. In the **Entity type** box, type the name for an entity type. For example, models sometimes use "Person" as an entity type to characterize people in an organization.
3. Under **Configurations**, click to choose an input field that identifies the entity type.
4. Under **Properties**, in the **Update mode** drop-down list box, choose how to update existing property values.

For more information, see [Update mode](#) on page 136.

Note: This setting is unavailable when **Write mode** is set to Initial load. For more information, see [Write mode](#) on page 134.

5. Under **Properties**, click in the **Add property** drop-down list box to select input fields to define properties for each entity configuration. By default the property name will be set to the field name. To change the name for any property, enter the new name in **Property name** box.
6. To add additional configurations to the model, repeat steps [3](#) on page 130 and [5](#) on page 130.

You can use more than one configuration to connect an entity type to itself (for example, Person reportsTo Person) or create or update an entity type based on two different fields, either from the same record or records from different sources.

Input source may include an "Name" field to identify a person type and a "Manager" field to identify a person type.

7. Click **OK**.

Configure an existing entity type

This procedure describes how to configure entity types from an existing model in the Write to Model stage.

Note: This procedure describes how to update entities from an existing Context Graph model. To configure a new entity type, see [Configure a new entity type](#) on page 130.

1. In the **Entity Types** box, drag an existing entity type icon to the canvas. Completing this step expands the **Entity Configuration** panel, which shows the selected entity type in the **Entity type** box.
2. Under **Configurations**, click to choose an input field that identifies the entity type. Input source may include an "Employee" field to identify workers in an organization, so an "Employee" configuration could identify a "Person" entity type.
3. Under **Properties**, in the **Update mode** drop-down list box, choose how to update existing property values.

For more information, see [Update mode](#) on page 136.

Note: This setting is unavailable when **Write mode** is set to Initial load. For more information, see [Write mode](#) on page 134.

4. Under **Properties**, click in the **Source field** column to associate an input field with each **Property name**. Leave this set to **None** when there is no corresponding source field for a property name in a configuration.
5. To add additional configurations to the model, repeat steps **2** on page 131 through **4** on page 131. You can use more than one configuration to connect an entity type to itself (for example, Person reportsTo Person) or create or update an entity type based on two different fields, either from the same record or records from different sources. Input source may include an "Name" field to identify a person type and a "Manager" field to identify a person type.
6. Click **OK**.

Configure new relationships

1. On the canvas, click and drag between entities to create a relationship. The order in which you click and drag defines the relationship. Completing this step expands the **Relationship Configuration** panel.

2. If the input source contains a field with the relationship label, set **Relationships provided at runtime** to **YES**.
3. In the **Relationship label** box, type the name of the relationship label.
If you set **Relationships provided at runtime** to **YES** in the previous step, in the **Relationship label field** drop-down list box, click the input field that contains the relationship label.
4. In the **Connections** drop-down list box, click the connection that you want to use for the relationship.
The list box shows a connection for each configuration defined for an entity type.
5. If it applies, check the **Allow more than one relationship based on a unique ID** check box, and select the input field that contains the unique ID.
6. On the **Properties** tab, in the **Update mode** drop-down list box, choose how to update existing property values.
For more information, see [Update mode](#) on page 137.

Note: This setting is unavailable when **Write mode** is set to Initial load. For more information, see [Write mode](#) on page 134.

7. On the **Properties** tab, in the **Source field** column, click the **Add property** drop-down list box to define properties for the relationship connection.
To change the name for any property, enter the new name in **Property name** column.
8. Optional: On the **Conditions** tab, you can add conditions that control whether to create a relationship between a source and a target entity.
For more information, see [Conditions](#) on page 137.
9. To add additional connections, repeat steps [4](#) on page 132 through [7](#) on page 132.
10. Click **OK**.

Configure existing relationships

1. On the canvas, click and drag between entities to create a relationship.
The order in which you click and drag defines the relationship.
Completing this step expands the **Relationship Configuration** panel.
2. In the **Relationship label** box, select an existing relationship label from the drop-down list.
3. In the **Connections** drop-down list box, click the connection that you want to use for the relationship.
The list box shows a connection for each configuration defined for an entity type.
4. If it applies, check the **Allow more than one relationship based on a unique ID** check box.
5. On the **Properties** tab, in the **Update mode** drop-down list box, click to choose when to update properties.
For more information, see [Update mode](#) on page 137.

Note: This setting is unavailable when **Write mode** is set to Initial load. For more information, see [Write mode](#) on page 134.

6. On the **Properties** tab, click in the **Source field** column to associate input fields with each property listed in the **Property name** column.
7. Optional: On the **Conditions** tab, you can add conditions that control whether to create a relationship between a source and a target entity.
For more information, see [Conditions](#) on page 137.
8. To add additional connections, repeat steps [3](#) on page 132 through [6](#) on page 133.
9. Click **OK**.

Reference

Stage ports

The Write to Model stage supports one input channel and one optional output channel.

Input

The Write to Model stage requires that your dataflow contain at least one source stage with defined fields that you can use to create a model.

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.

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.

Write to Model options

The Write to Model options allow you to create a new model or select an existing model to update. Other options on this page allow you to clear an existing model, remove orphaned entities, and index fields.

- | | |
|-------------------|--|
| Model | Specifies the Context Graph model to which to load data. To create a new model enter the name for the model. To update an existing model, click to select the model name from the drop-down list. |
| Write mode | Specifies whether this is an initial load, exclusive lock, or concurrent lock. <ul style="list-style-type: none"> • Initial load—Choose this mode if you are loading data into model for the first time. The model will be locked so that other Context Graph stages will be unable to read from or write to the model. The Properties Update mode will be greyed out and unavailable in the entity and relationship configurations. Selecting this option checks |

and greys out the **Clear model** check box, so that any existing data will always be cleared prior to writing.

This mode provides better performance when initially loading a model. If you have multiple input files in your dataflow, you can input fields from all of them to create properties, but you cannot use them to update existing properties with new values.

- **Exclusive lock**—Choose this mode to prevent other Context Graph stages from concurrently writing to a model. This mode allows a model to be cleared and for existing model properties to be updated.
- **Concurrent writes**—Choose this option to allow the model to be concurrently read from and written to by other Context Graph stages.

Clear model Check this check box to remove all existing entities and relationships before processing new data. Uncheck this option to update existing entities and relationships with any new information.

Note: Selecting this check box does not alter security settings for the Context Graph. The model will be recreated, but the security settings will remain the same.

Remove orphaned entities Check this check box to remove entities from the model that have no relationships upon completing the Write to Model operation.

Entity types This panel provides a palette of entity types. For a new model, the panel only shows the **New Entity Type**. For an existing model, the panel also shows entity types from the model metadata. Drag a **New Entity Type** onto the canvas to define a new entity type or drag an existing entity type onto the canvas to map its properties to input fields.

Indexes The **Indexes** button is available after you define at least one entity with at least one property. Click this button to expand the **Indexes** panel to index fields. For more information about options on the **Indexes** panel, see [Indexes](#) on page 134.

Show schema This **Yes/No** toggle switch displays the schema for an existing model. The schema provides a guide to add entities and relationships to the Write to Model stage. The toggle switch appears only after you select an existing model in the **Model** box. By default, it is set to **No**. If you click to toggle this switch to **Yes**, the schema for the selected model appears as dimmed entities and relationships on the canvas. You can then click dimmed entities to configure and add them to the stage. Entities appear undimmed after they are configured. Relationships between entities can be configured in the same manner after you configure both the source and the target entity for any relationship.

Indexes

Select entity properties to index on this panel and specify whether to apply case sensitive (exact) or case insensitive indexing for the selected entities.

This panel is expanded when you click the **Indexes** button on the Write to Model stage canvas. Indexing properties results in faster response times when the fields are queried, and some search options work only on indexed fields. Unindexed properties may be queried, but response times will

be slower. On the other hand, indexing every field in a model reduces the performance when writing data to the model. To maximize overall performance, one can index fields likely to be searched and not index fields that are less likely or unlikely to be queried.

Property	This column shows entity properties defined in the model.
Type	This column shows the data type for a property.
Index Type	In this column, choose whether to index a property. There are three different options. <ul style="list-style-type: none"> • None—Choose this option when a property is unlikely to be queried often. • Exact—Search results must exactly match upper and lower case characters in the indexed data. Choose this option to index when for more precise search results. • Case Insensitive—Search results do not have to match upper and lower case characters in the indexed data. A case insensitive index typically returns more results than an exact search.

Entity Configuration

The **Entity Configuration** panel allows you to define one or more configurations to connect an entity type to itself, (such as `Person reportsTo Person`) or to create or update an entity type based on two different fields. These can be either from the same record or from records from different sources. You can specify a name for the new entity type and add one or more configurations. Each entity configuration can have its own set of properties.

Entity type	Specifies the entity name.
Configurations	Configurations map input fields to entity types (as in <code>Person:Name</code>). These mappings create unique identifiers for each Entity. This identifier combines the Entity Type and Entity Label (data from the input field) to create a unique identifier called the Entity ID (for example, <code>Person:Churchill</code>). This mapping provides the source and target identifiers for relationships (as in <code>Person:Name → Event:EventName</code>).

Properties

For each configuration, this section maps property names to source fields. The mapping depends on whether you are defining a known entity type or a new entity type. For a known entity type, you map source fields to the property names. For a new entity type, you choose source fields and enter names to define properties.

Update mode	Choose how Write to Model manages property updates. After an entity is created it can be updated over time when data with the same Entity ID is input into the Write to Model stage. <ul style="list-style-type: none"> • 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 non-empty properties**—Properties are always updated unless the input is a null or empty string.
- **Never overwrite properties with empty input data**—Properties are never updated once populated with non-empty data.

Property name	Specifies the name of an entity property. For a known entity type, this column is populated with property names from the entity. For a new entity type, enter a property name in this column for each selected source field. By default, the property name is the source field name.
Source field	Specifies the source field of an entity property. For a known entity type, select a source field that maps to a property. For a new entity type, select the source fields in this column that will populate entity properties. Choose None if there is no source field that maps to a property.

Relationship Configuration

The **Relationship configuration** determines how relationships are created between source and target entities.

You can specify a name for the new relationship label and add one or more connections, or you can select a relationship label from an existing model. Connections are determined by the configurations of the source and target entities.

Relationships provided at runtime	This setting allows you to specify whether the input source contains a field with the relationship label. <ul style="list-style-type: none"> • YES—Select this option if your input source contains a field with the relationship label. • NO—Select this option to specify the relationship label here.
Relationship label	Displayed when Relationships provided at runtime is set to NO . Specify the name for the relationship label. For an existing model, you can choose a relationship from the drop down list.
Relationship label field	Displayed when Relationships provided at runtime is set to YES . Select a field from the input source that contains the relationship label.
Connections	Shows relationships between the source and target entities. Choose source and target identifiers that you want to include in the relationship. Connections are defined by the source and target entity configurations.
Allow more than one relationship based on unique ID	To allow a relationship to be created more than once between a source and target entity, check this check box. Then in the drop-down list box, select the source field on which to base the relationship.



Properties

For each connection, options on this tab map property names to source fields. The mapping depends on whether you are defining properties for a known relationship label or a new relationship label. For a known relationship label, you map source fields to the property names. For a new relationship label, you select source fields and enter names to define properties.

Update mode	<p>Choose how Write to Model manages property updates. After a relationship is created it can be updated over time when data with the same Relationship ID is input into the Write to Model stage.</p> <ul style="list-style-type: none"> • 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 non-empty properties—Properties are always updated unless the input is a null or empty string. • Never overwrite properties with empty input data—Properties are never updated once populated with non-empty data. <p>These settings are unavailable when Write mode is set to Initial load. For more information, see Write mode on page 134.</p>
Property name	Specifies the name of a relationship property. For a known relationship, this column is populated with property names from the relationship. For a new relationship, enter a property name in this column for each selected source field. By default, the property name is the source field name.
Source field	Specifies the source field of a relationship property. For a known relationship, select a source field that maps to a property. For a new relationship, select the source fields in this column that will populate relationship properties. Choose None if there is no source field that maps to a property.

Conditions

For each connection settings on this tab specify conditions that control whether to create a relationship between a source and a target entity.

Add Conditions	Click this button to add conditions to a relationship.
Logical operator	This field is visible when you add a second or succeeding condition. Click And or Or to specify how a condition is applied in combination with preceding conditions. The And operator displays a record if a condition and preceding condition are both true. The Or operator displays a condition when either the condition or the preceding condition are true. A relationship is added if the entire sequence of conditions evaluates to true.
Data source	Select whether a condition applies to an Input field , a Source property (on the source entity), or a Target property (on the target entity).
Name field	Click to select the field or property name to which the condition applies.
Operator	Click to select the operator that defines the condition.
Value	Enter the condition value acted on by the operator.
Add condition button 	Click this button to add a condition.
Delete condition button 	Click this button to delete a condition.

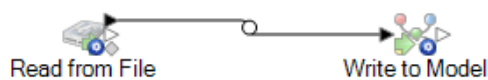
Example Context Graph Dataflows

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.

Write to Model with flat file input

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:

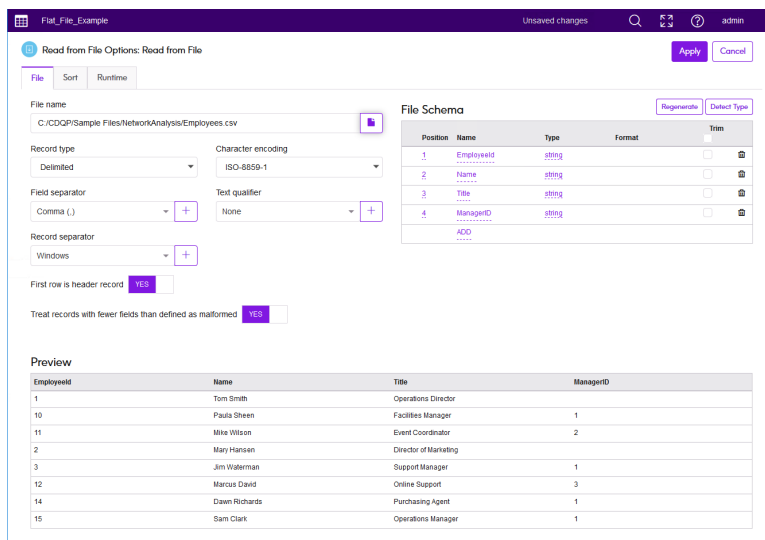
- Employee ID
- Name
- Title
- Manager ID

The input file itself looks like this:

```
F
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 options appear as follows when the stage 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 is drag **New Entity Type** to the canvas. In the **Entity type** box, we type "Employee". Under **Configurations** we select "Employeeid". Under properties we add Name and Title as we want those fields brought in as properties for the Employee entity type. In each case we leave the **Property name** set to the same as the **Source field**.

Entity Configuration

Entity type

Employee

Configurations



EmployeeId



Add input field

Properties

Update mode

Never overwrite properties with empty input data

Source field

Property name

Name

Name



Title

Title



Add property

OK

Cancel

In the **Update mode** box, we configure the processing options for the configuration. This option specifies 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 the first time that it processed Mary Hansen. 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 in the first instance and missing in a subsequent instance. This also ensures that the order in which these records are read has no impact on the model.

Properties

Update mode

Never overwrite properties with empty input data

Always update properties

Update properties unless all input is null

Never overwrite non-empty properties

Never overwrite properties with empty input data

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. In this case we select the default, **Never overwrite properties with empty input data**.

We repeat these steps to "ManagerId" as a second configuration. Although ManagerID and EmployeeID are different fields in the input file, both are configurations of "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.

Entity Configuration

Entity type

Employee

Configurations



EmployeeId



ManagerID



Add input field

Properties

Update mode

Never overwrite properties with empty input data

Add property

OK

Cancel

Delete

We click **OK** to save this entity type. This displays the Employee entity type on the canvas.

Now we configure relationships. The first thing we do is click the Employee entity type and drag a relationship line to itself. This displays the **Relationship Configuration** panel. The relationship between entities reflects the reporting structure (employee to manager). In the **Relationship label** box, we enter the text, "reports_to". Under connections, we select "Employee:EmployeeId->Employee:ManagerID". This defines the EmployeeID as the source and ManagerID as the target of a relationship. We could reverse the relationship by specifying "manages" instead of a "reports_to", in which case we would choose the reverse permutation of source and target fields. Once again we choose to **Never overwrite properties with empty input data**.


The completed Relationship Configuration is as follows:


Entity Configuration


Entity type

Employee

Configurations

EmployeeId 


ManagerID 



Properties

Update mode

Never overwrite properties with empty input data 

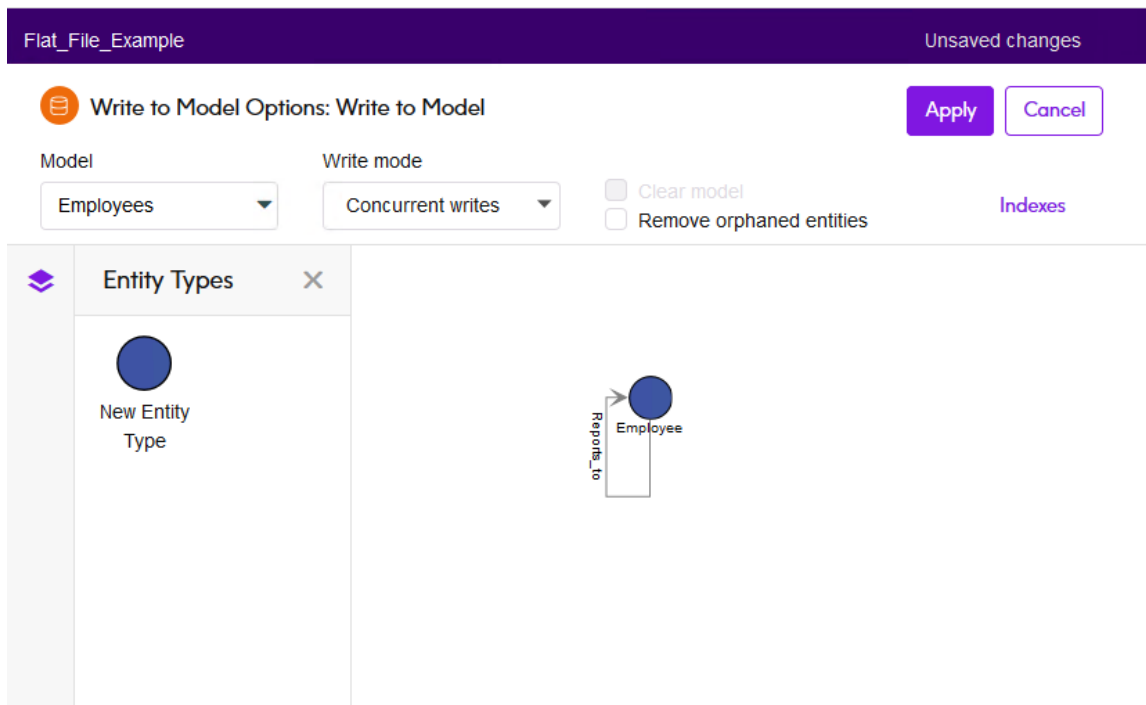
Add property 

OK


Cancel

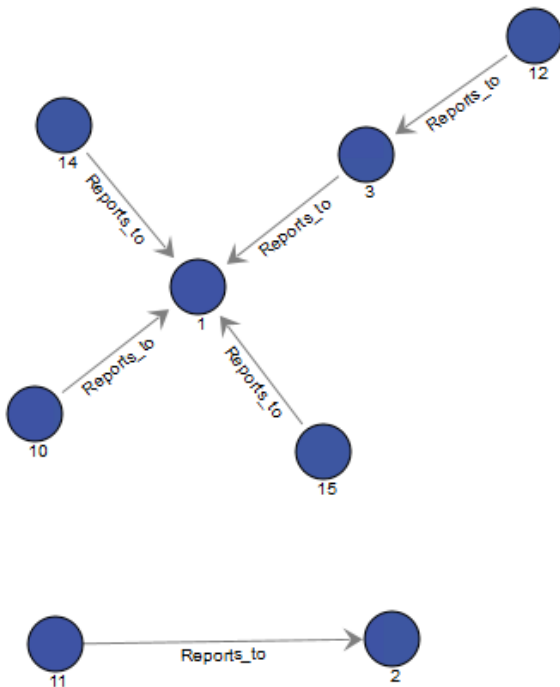
Delete


The configuration of this dataflow is complete and results in the following model, as depicted on the canvas.



We click **Apply** to save the model.

When we run the flow, this results in the following graph database as viewed on the **Context Graph Visualization Canvas** . On the Canvas you can view properties by clicking an entity.

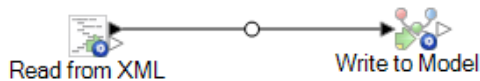


Another way to view this same data is in the **Context Graph Visualization Table View** . In Table View you can view property values simultaneously for multiple entities.

<input type="checkbox"/>	Entity	Entity ID	Name	Title
<input type="checkbox"/>	1	Employee:1	Sam Clark	Operations Manager
<input type="checkbox"/>	10	Employee:10	Paula Sheen	Facilities Manager
<input type="checkbox"/>	11	Employee:11	Mike Wilson	Event Coordinator
<input type="checkbox"/>	12	Employee:12	Marcus David	Online Support
<input type="checkbox"/>	14	Employee:14	Dawn Richards	Purchasing Agent
<input type="checkbox"/>	15	Employee:15	Sam Clark	Operations Manager
<input type="checkbox"/>	2	Employee:2	Mary Hansen	Director of Marketing
<input type="checkbox"/>	3	Employee:3	Marcus David	Online Support

Write to Model with XML input

The Write to Model dataflow that uses an XML file for input looks like this:

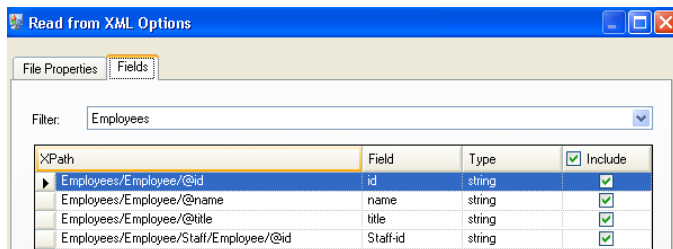
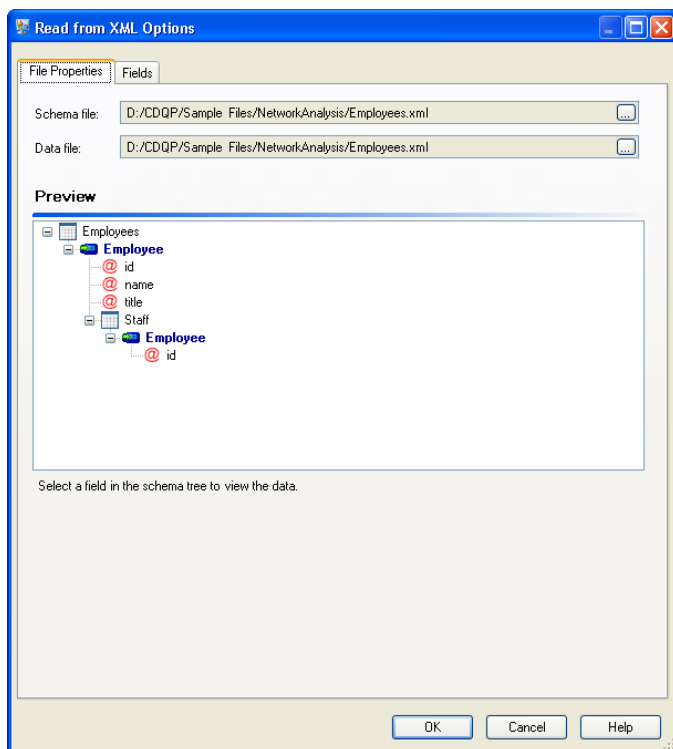


Configuring Read from XML

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>
```

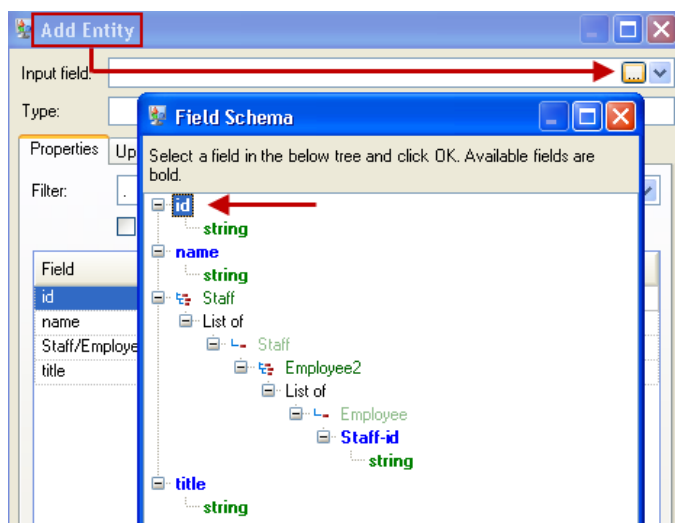
The Read from XML stage appears as follows when it is configured to work with this input file:



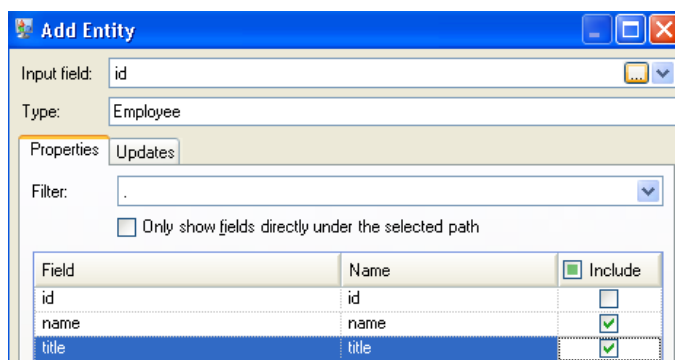
Configuring Write to Model

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.

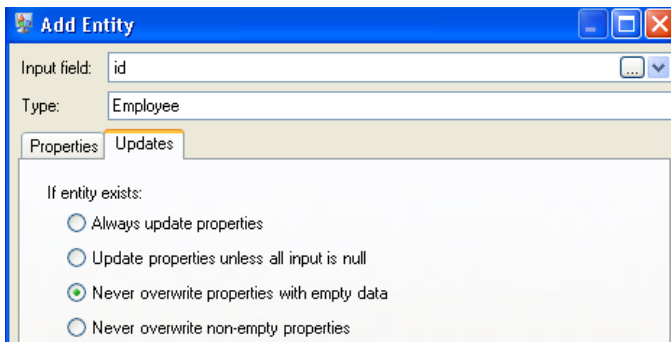
Because we are creating a model that is similar to an organization chart, our entities are employees who are assigned numeric IDs. The first thing we do on the **Add Entity** dialog box is click the browse button to access the **Field Schema** dialog box, and then select "id." This is the first group 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.



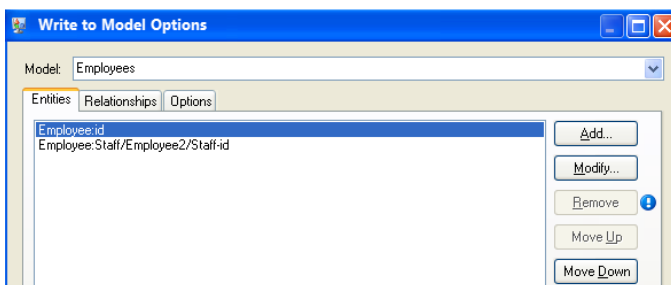
After setting properties for the ID entity, we configure the processing options. The Updates tab enables you to specify whether properties can be updated in the model once they are in place and if they should overwrite existing data. For instance, in our example, Mary Hansen would be encountered twice 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.

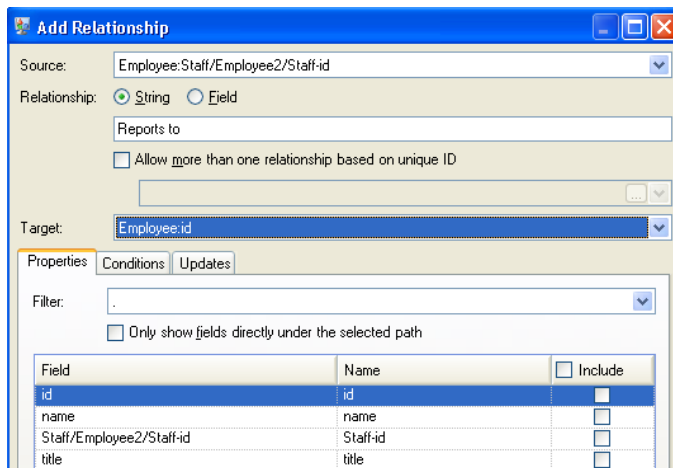
We repeat these steps to add "ManagerId" as the second group of entities in our model. Although ManagerID and EmployeeID are different fields in the input file, both entities' types are set to "Employee." If we set ManagerID to a different type, the model would contain two entities for mid-level managers. 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:

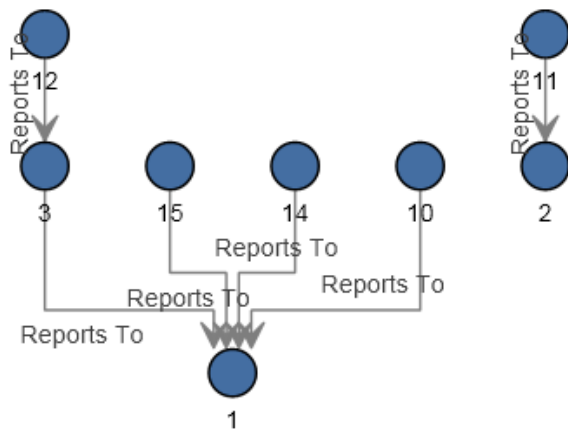


Now we configure the Relationships tab. The first thing we do on the **Add Relationship** dialog box is select the source of the relationship from the list of entities created on the Entities tab. The relationship between our entities reflects the reporting structure (employee to manager); therefore, 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.

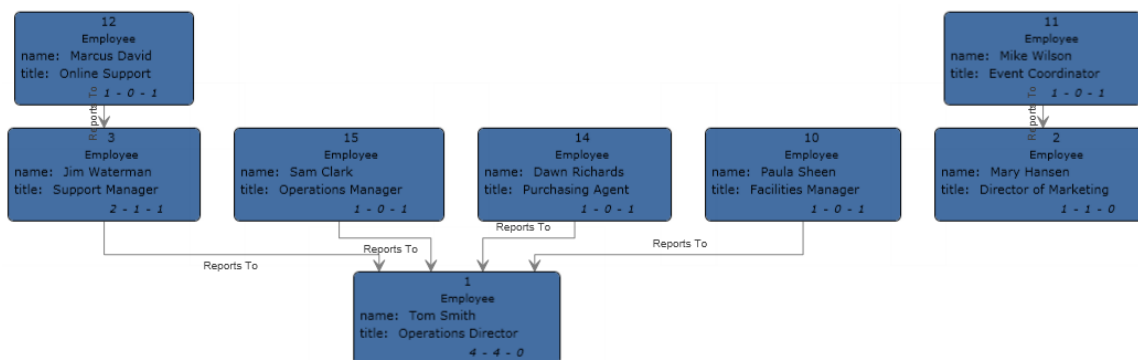
The completed Relationships tab for this example appears as follows:



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 Stewardship

Data Stewardship stages

These stages are available with Data Stewardship.

Stage	Icon	Description
Exception Monitor on page 150		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

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 addition to setting conditions that determine if records require manual review, you can also configure Exception Monitor to send a notification to one or more email addresses when those conditions have been met a certain number of times.

How to

Reference

Stage ports

The Exception Monitor stage has a single input channel and two output channels.

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 collects records that do not meet any of the conditions defined in the Exception Monitor stage. The exception port, contains all records that match one or more exception conditions. The exception port may also include non-exception records if you enable **Return all records in exception's group**. Exception Monitor does not add or modify fields within a record.





Options tab


Settings tab

Enterprise Data Integration




These stages are available to you if you have licensed for the Enterprise Data Integration module. Use these to connect to data in multiple sources either directly or through integration with your existing data access technologies.



Stage	Icon	Description
DB Loader	 DB Loader	The DB Loader stage allows you to access and load data from and to the configured databases. This stage provides an interface to a high-speed data loading utility. Currently, the platform supports Oracle Loader, DB2 Loader, PostgreSQL Loader, and Teradata Loader
Field Selector	 Field Selector	Use the Field Selector stage to choose the fields that are to be passed to the next stage in the dataflow and remove the unwanted ones. 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 the Field Selector to retain only the new field and remove the two source fields from the dataflow.
Generate Time Dimension	 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 for time period.
Query DB	 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.

Stage	Icon	Description
Read from DB	 Read from DB	<p>The Read From DB stage reads data from a database table/view as input to a dataflow. The stage is available for jobs, services, and subflows but not for process flows.</p> <p>Note: The stage supports reading data from and writing data to HDFS 3.x and Hive 2.1.1. The support includes:</p> <ul style="list-style-type: none"> • 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 <p>Also see Best Practices for connecting to HDFS 3.x and Hive 2.1.1.</p>
Read from Spreadsheet	 Read from Spreadsheet	<p>The Read from Spreadsheet stage reads data from a spreadsheet, for example, an Excel spreadsheet, as input to a dataflow. It supports these two formats:</p> <ul style="list-style-type: none"> • .xls • .xlsx
SQL Command	 SQL Command	<p>SQL Command runs one or more SQL commands for each record in the dataflow. You can use SQL Command to:</p> <ul style="list-style-type: none"> • Run 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.
Write to Spreadsheet	 Write to Spreadsheet	<p>The Write to Spreadsheet stage writes data to a spreadsheet, such as an Excel spreadsheet, as output from a dataflow.</p>

Stage	Icon	Description
Write to DB	 Write to DB	<p>The Write to DB stage writes the output of a dataflow to a database. The stage writes all values of the <code>date</code> datatype as <code>String</code> values. This is the behavior of the <i>JTDS driver</i>, which is the default driver used by Spectrum. To handle all <code>date</code> datatype values as is, use Microsoft's JDBC driver.</p> <p>Note: The stage supports reading data from and writing data to HDFS 3.x and Hive 2.1.1. The support includes:</p> <ul style="list-style-type: none"> • 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 <p>Also see Best Practices for connecting to HDFS 3.x and Hive 2.1.1.</p>

Data Quality

Stage	Icon	Description
Filter	 Filter	<p>The Filter stage retains or removes records from a group of records based on the rules you specify.</p>
Match Key Generator	 Match Key Generator	<p>This stage creates a match key for each record. A match key is a non-unique key shared by like records that identify records as potential duplicates. The match key facilitates the matching process by only comparing records that contain the same match key.</p>
Write to Search Index	 Write to Search Index	<p>The Write to Search Index stage enables you to create a full-text index based on the data coming into 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.</p>

Stage	Icon	Description
Intraflow Match	 Intraflow Match	The Intraflow Match stage 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.
Best of Breed	 Best of Breed	The Best of Breed stage consolidates duplicate records by selecting the best data in a duplicate record collection and creating a new consolidated record using the best data.

Machine Learning

Binning

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*.

How to

Add Binning to workflow

1. In the **Stages** panel, scroll to **Machine Learning**, and drag the **Binning** stage onto the canvas.

2. Connect the stage to other stages.
For more information, see [Binning Stage ports](#).
3. Double-click the Binning stage to open the **Binning Properties**.
4. On the **Binning Properties** tab, configure the model name and the input fields to be included in the binning.
For more information about options on this tab, see [Binning Properties tab](#) on page 156.
5. On the **Basic Options** tab, configure the binning style, null value bin, number of target bins, and bin width.
For more information about options on this tab, see [Basic Options tab](#) on page 156.
6. Click **Apply** to save your changes.

Reference

Stage ports

Input

The input stage must be the data source that contains input variable fields for your model.

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

An output stage is not required. You may connect an output stage if you wish to capture your output independent of the Machine Learning Model Management tool.

Binning Properties tab

Model Name	Specifies the name of a binning model.
Overwrite	Check this check box to overwrite data in an existing model.
Description	Provides space to document the purpose of a model.
Inputs	This table shows numeric input fields along with the data type. Check the Include check box to include data from a field in binning. Note: Only numeric fields appear in this list.

Basic Options tab

Binning Style	Select whether to perform an Equal-Range or Equal Population binning.
Null value bin	Specifies how to handle empty bin fields. These represent unknown values due to missing data. <ul style="list-style-type: none"> • Highest—Assigns null values to the highest bin • Lowest—Assigns null values to the lowest bin. <p>The lowest bin is always bin 1.</p>
Target internal bins	Specifies the number of bins 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.
Bin width	Choose this option to perform equal-range binning. 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.

Logistics Regression

Logistic Regression enables you to perform machine learning by creating models from datasets that use binary objectives with input variables.

How to**Add Logistics Regression to workflow**

This procedure describes how to add the Logistics Regression stage to a workflow in Flow Designer. Logistics Regression is a Machine Learning stage.

To create your model, you must first complete the **Model Properties** settings. The **Basic Options** and **Advanced Options** settings provide sufficient default settings to complete a job, but you can change those settings to meet your needs. After you run your job 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.

1. In the **Stages** panel, scroll to **Machine Learning**, and drag the **Logistics Regression** stage onto the canvas.

2. Connect the stage to other stages.

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.

For more information, see [Logistics Regression stage ports](#).

3. Double-click the Logistics Regression stage to open **Logistic Regression Options: Logistics Regression**.
4. On the **Model Properties** tab, configure the model name, number of principal components, and the input fields to be included in the analysis.

For more information about options on this tab, see [Model Properties tab](#) on page 158.

5. On the **Basic Options** tab, configure to use all factor level, to score input data, the transform, and how to handle missing data.

For more information about options on this tab, see [Basic Options tab](#) on page 158.

6. On the **Advanced Options** tab, configure whether to ignore constant fields, the PCA method, and convergence criteria.

For more information about options on this tab, see [Advanced Options tab](#) on page 159.

7. On the **Model Output** tab, view the metrics you are using to assess the fitted model.

For more information about this tab, see [Model Output tab](#) on page 161

8. Click **Apply** to save your changes.

Reference

Stage ports

Input port

The input stage must be the data source that contains the principal components for your model.

Output ports

Output is captured by the Machine Learning Model Management tool. The optional output ports allow you to pass output to subsequent stages in a workflow.

Model score port Use this port to capture model scores independent of the Machine Learning Model Management tool.

Model metrics port The optional model metrics port lets you output the model assessment metrics to a data file. This helps compare many models generated from within and outside of Spectrum Technology Platform and perform other data processing

tasks on the metrics. This port's functionality is determined by input and configuration of basic and advanced options in the stage settings.

Model Properties tab

Options must be configured on this tab to perform an analysis.

Model name	You can enter a custom name for the model to use as a reference. By default, Spectrum automatically generates a name.
Overwrite	Check this check box to overwrite an existing model with new data.
Objective field	The field for objective function values used for learning.
Description	Provides space to describe the model in a workflow.
Inputs	<p>This table shows input fields along with the data type and model data type. Check the Include check box to include data from a field in the model. In the Model Data Type column, specify whether an input field is a categorical, datetime, numeric, string, or uniqueid field.</p> <p>In the Model Data Type, click the drop-down list to specify whether each input field is to be used as a numeric, categorical, or datetime field.</p>

Basic Options tab

Standardize input fields	<p>Check the check box to standardize the numeric columns to have zero mean and unit variance. This is the default.</p> <p>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.</p>
Score input data	Check this check box to add a column for the model prediction (score) to the input data.
Prior	Check this check box if the data has been sampled and the mean of response does not reflect reality. Enter the prior probability for $p(y=1)$ in the text field. The default value is 0.5.
Missing data	<p>This option specifies how to handle missing data.</p> <ul style="list-style-type: none"> • Skip—Skips missing data. • Impute means—Adds the mean value for missing data.
Sampling	<ul style="list-style-type: none"> • Percentage for training data—Specify a value between 1 and 100 when the input data is randomly split into training and test data samples. • Percentage for test data—Enter the value of 100 minus the value entered in Percentage for training data. • Seed for sampling—Enter a number to ensure that when the data is split into test and train data in the same way each time you run the dataflow. Uncheck this field to get a random split each time you run the flow.

Advanced Options tab*Options*

Ignore constant fields	Leave this check box checked to skip fields that have the same value for each record.
Compute p values	Leave this check box checked to calculate p values for the parameter estimates
Remove collinear column	Leave this check box checked to automatically remove collinear columns during model building. This option must be checked if Compute p values is also checked. This results in a 0 coefficient in the returned model.
Include constant term (Intercept)	Leave this check box checked to include a constant term (intercept) in the model. This field must be checked if the Remove collinear column check box is also checked.
Solver	Select a solver from in the drop-down list box. <ul style="list-style-type: none"> • 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. • LBFGS— Ideal for datasets with many columns.

Note: **CoordinateDescent** and **CoordinateDescentNaive** are currently experimental.

Convergence Criteria

Maximum iterations	Specifies the number of training iterations that should take place.
Objective epsilon	Specifies the threshold for convergence. If the objective value is less than this threshold, the model will be converged. This must be a value between 0 and 1, exclusive. The default setting is 0.0001.
Beta epsilon	Specifies the threshold for convergence. 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. This must be a value between 0 and 1, exclusive. The default setting is 0.0001.

Cross Validation

Seed for N fold	Leave this check box checked and enter a seed number to ensure that when the data is split into test and train data in the same manner each time you run the
------------------------	--

dataflow. Uncheck this field to get a random split each time you run the flow. The default setting is 15341.

- N fold** Check this check box and enter the number of folds to perform cross validation.
- Fold assignment** Check this check box and select from the drop-down list if you are performing cross-validation. This field is applicable only if you entered a value in the **N fold** box and the **Fold field** is not specified.
- **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.
- Fold field** If you are performing cross-validation, check this check box 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**.

Regularization

- Regularization type** Choose the appropriate regularization type. 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.
- **Elastic Net Penalty**—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.
 - **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.

Value of alpha	<p>Check this check box 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.</p> <p>Note: A single equals sign is an assignment operator meaning "is," while the double equals sign is an equality operator meaning "equal to."</p>
Value of lambda	<p>Check this check box 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.</p>
Search for optimal value of lambda	<p>Check this check box 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.</p> <ul style="list-style-type: none"> • Stop early if no relative improvement—Check this check box to end processing when there is no more relative improvement on the training or validation set. • Maximum lambda search—Check this check and enter the maximum number of lambdas to use during the process of lambda search.
Maximum active predictors	<p>Check this check box 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</p>

Model Output tab

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.

Principal Component Analysis (PCA)

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 default settings to complete a job, but you can change those settings to satisfy particular circumstances. 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.

How to

Add Principal Component Analysis (PCA) to workflow

1. In the **Stages** panel, scroll to **Machine Learning**, and drag the **PCA** stage onto the canvas.
2. Connect the stage to other stages.

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.

For more information, see [PCA Stage ports](#).

3. Double-click the PCA stage to open **PCA Properties**.
4. On the **Model Properties** tab, configure the model name, number of principal components, and the input fields to be included in the analysis.

For more information about options on this tab, see [Model Properties tab](#) on page 163.

5. On the **Basic Options** tab, configure to use all factor level, to score input data, the transform, and how to handle missing data.

For more information about options on this tab, see [Basic Options tab](#) on page 163.

6. On the **Advanced Options** tab, configure whether to ignore constant fields, the PCA method, and convergence criteria.

For more information about options on this tab, see [Advanced Options tab](#) on page 163

7. Click **Apply** to save your changes.

Reference

Stage ports

Input port

The input stage must be the data source that contains the principal components for your model.

Output ports

Output is captured by the Machine Learning Model Management tool. The optional output ports allow you to pass output to subsequent stages in a workflow.

Model score port Use this port to capture model scores independent of the Machine Learning Model Management tool.

Model metrics port The optional model metrics port lets you output the model assessment metrics to a data file. This helps compare many models generated from within and

outside of Spectrum Technology Platform and perform other data processing tasks on the metrics. This port's functionality is determined by input and configuration of basic and advanced options in the stage settings.

Model Properties tab

Options must be configured on this tab to perform an analysis.

Model name	You can specify a custom name for the model to use as a reference. By default, Spectrum automatically generates a name.
Overwrite	Check this check box to overwrite an existing model with new data.
Principal components	Enter the number of principal components that you want your model to contain.
Description	Provides space to describe the model in a workflow.
Inputs	This table shows input fields along with the data type and model data type. Check the Include check box to include data from a field in the model. In the Model Data Type column, specify whether an input field is a categorical, datetime, numeric, string, or uniqueid field.

Basic Options tab

User all factor level	Specifies whether to retain the first principal component, which has the largest variance in the data. <ul style="list-style-type: none"> • Check this option to retain the first principal component. • Uncheck this option to skip the first principal component. This is the default.
Transform	Specify the transformation method for numeric columns in the training data. <ul style="list-style-type: none"> • 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.
Missing data	Specifies whether to impute missing entries with the column mean. <ul style="list-style-type: none"> • Skip—Choose this option to skip missing data. This is the default setting. • Impute mean—Choose this option to add the mean value for any missing data.

Advanced Options tab

Ignore constant fields	Check this check box to skip fields that have the same value for each record, since no information can be gained from them. This option is checked by default.
PCA Method	Specify the algorithm to use for computing the principal components:

- GLRM—Fits a generalized low-rank model with L2 loss function and no regularization; solves for the SVD using local matrix algebra.
- GramSVD—Uses a distributed computation of the Gram matrix, followed by a local SVD using the JAMA package. This is the default method.
- Power—Computes the SVD using the power iteration method.
- Randomized—Uses the randomized subspace iteration method.

Maximum iteration Specifies the number of training iterations. The value must be between 1 and 1e6 and the default is 1000.

Output tab

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.

5 - File properties reference

In this section

Supported character encoding methods.....	166
Field separators.....	166
Record separators.....	167



Supported character encoding methods

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 separators

Use one of these characters to separate fields in a delimited file.

- Space
- Tab
- Comma
- Period

- Semicolon
- Pipe

Record separators

Specifies the character used to separate records in line a sequential or delimited file. Note that this selection is not available if you select **Use default EOL**.

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.

Note: If your file uses a different record separator, select another character as a record separator.

6 - Supported data types reference

Depending on the type of processing you want to perform you may use one or more of . 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.

In this section

Data types.....	169
-----------------	-----



Data types

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.

7 - Date and time patterns reference

In this section

Date and time patterns.....171



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

Letter	Description	Example
d	Day of the month	10
dd	Two-digit day of the month. Numbers less than 10 have a zero added.	09
F	Day of the week in month	2
E	Short name of the day of the week	Tue
EEEE	Long name of the day of the week	Tuesday
a	AM PM marker	PM
H	Hour of the day, with the first hour being 0 and the last hour being 23.	0
HH	Two-digit hour of the day, with the first hour being 0 and the last hour being 23. Numbers less than 10 have a zero added.	08
k	Hour of the day, with the first hour being 1 and the last hour being 24.	24
kk	Two-digit hour of the day, with the first hour being 1 and the last hour being 24. Numbers less than 10 have a zero added.	02
K	Hour hour of the morning (AM) or afternoon (PM), with 0 being the first hour and 11 being the last hour.	0
KK	Two-digit hour of the day, with the first hour being 1 and the last hour being 24. Numbers less than 10 have a zero added.	02
h	Hour of the morning (AM) or afternoon (PM), with 1 being the first hour and 12 being the last hour.	12
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

Letter	Description	Example
mm	Two-digit minutes of the hour. Numbers less than 10 have a zero added.	05
s	Second of the minute	55
ss	Two-digit second of the minute. Numbers less than 10 have a zero added.	02
S	Millisecond of the second	978
SSS	Three-digit millisecond of the second. Numbers containing fewer than three digits will have one or two zeros added to make them three digits.	978 078 008
z	Time abbreviation of the time zone name. If the time zone does not have a name, the GMT offset.	PST GMT-08:00
zzzz	The full time zone name. If the time zone does not have a name, the GMT offset.	Pacific Standard Time GMT-08:00
Z	The RFC 822 time zone.	-0800
X	The ISO 8601 time zone.	-08Z
XX	The ISO 8601 time zone with minutes.	-0800Z
XXX	The ISO 8601 time zone with minutes and a colon separator between hours and minutes.	-08:00Z



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

www.precisely.com

© 2007, 2021 Precisely. All rights reserved.