SAS® Event Stream Processing 5.1: Using SAS® Event Stream Processing Studio

Overview to SAS Event Stream Processing Studio

Overview

SAS Event Stream Processing Studio is a web-based client that enables you to create, edit, upload, and test event stream processing models using SAS Event Stream Processing Studio Modeler. SAS Event Stream Processing Studio Modeler displays a model as a data flow diagram, enabling you to see and control how windows relate and flow into one another.

Requirements for Solution Access and Use

Here are the requirements for accessing and using SAS Event Stream Processing Studio:

- a supported web browser that has been installed

  Note: For more information about supported browsers in SAS Event Stream Processing Studio, click and then click **About**. Click **Supported Browsers and Platforms** to view supported browsers. SAS Event Stream Processing Studio requires the use of cookies to maintain the session state.
- a minimum screen resolution of 1,280 x 1,024
- JavaScript enabled in your browser

Starting the Event Stream Processing Server

Before opening or creating a model in SAS Event Stream Processing Studio, you must start the ESP server. To start the server on a UNIX system, run the following command:

```
$DFESP_HOME/bin/dfesp_xml_server -http port
```

2

On Windows systems, run the following command:

```
%DFESP_HOME%\bin\dfesp_xml_server -http port
```

In these examples, *port* specifies the port for the HTTP REST API. The terminal log should confirm successful server instantiation and reiterate the port specified here. For information about the ESP server, see "Using the ESP Server" in SAS Event Stream Processing: Using the ESP Server.

Accessing SAS Event Stream Processing Studio

If SAS Event Stream Processing Studio has been deployed with other SAS Viya applications, to access SAS Event Stream Processing Studio:

1. Open the following URL:

   `https://host/SASEventStreamProcessingStudio`

   The *host* is the system where SAS Event Stream Processing Studio is installed.

   The Sign In to SAS window appears.

   Note: The Sign In to SAS window appears only if your deployment has been configured to enable users to log on to SAS Event Stream Processing Studio. If your deployment has not been configured in this way, you are not required to enter a user ID and password to access SAS Event Stream Processing Studio.

2. Enter your user ID and password and click **Sign in**.

If you successfully access SAS Event Stream Processing Studio, the SAS Event Stream Processing Studio home page appears. If you are using SAS Event Stream Processing Studio for the first time, the initial SAS Event Stream Processing Studio window might not contain any models.

Alternatively, if SAS Event Stream Processing has been deployed as a single application, open the following URL:

`http://host:port/SASEventStreamProcessingStudio`

The *host* is the system where SAS Event Stream Processing Studio is installed. The *port,* which is provided to you during product configuration, is the HTTP port. By default, the HTTP port is 8080.

Understanding the User Interface

Pages

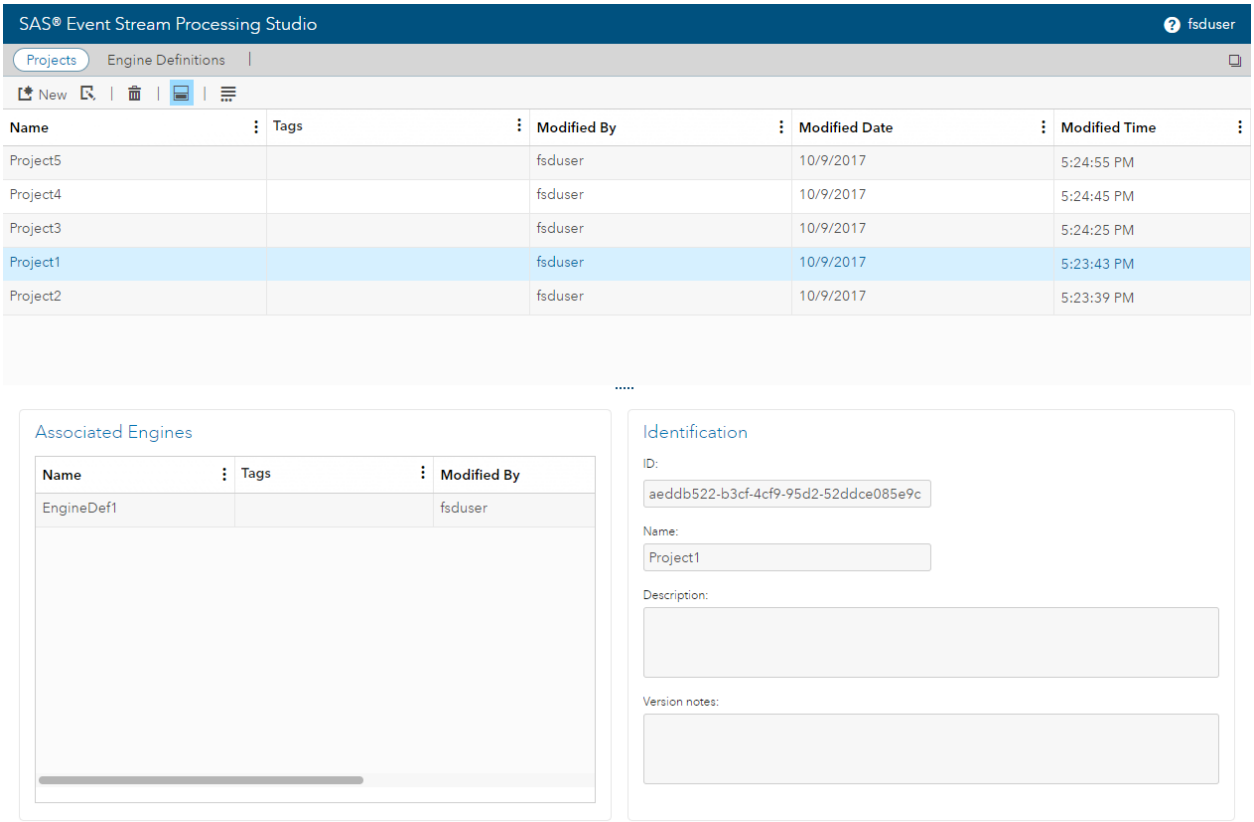
A *page* is the highest level container in the user interface. All other user interface elements are contained within the confines of a page.

SAS Event Stream Processing Studio contains the following main pages:

- the **Projects** page enables you to create, edit, upload, download, or delete the projects containing your SAS Event Stream Processing models
- the **Engine Definitions** page enables you to create, edit, upload, download, or delete engine definitions

When you first access SAS Event Stream Processing Studio, the **Projects** page appears.

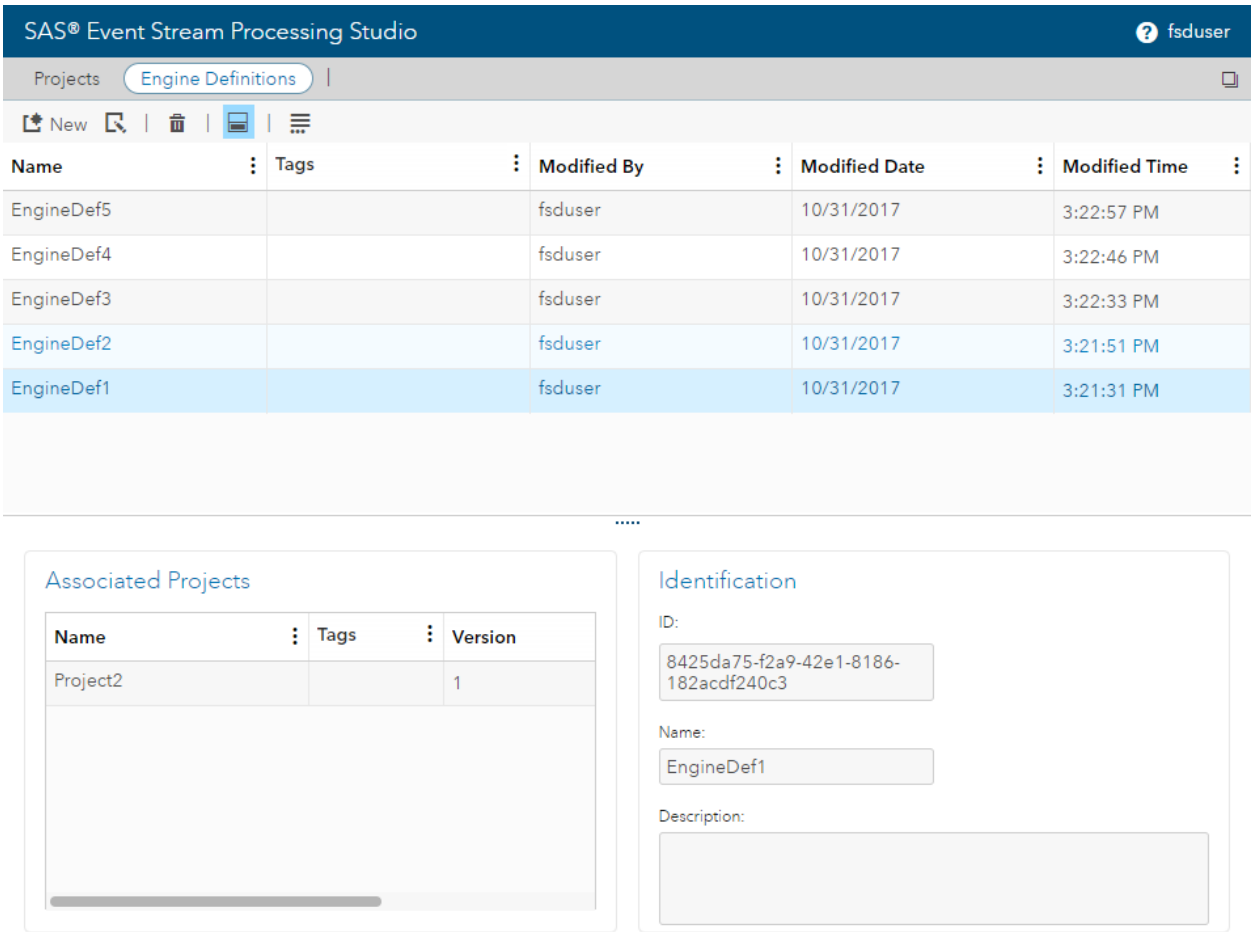
The Projects Page Displaying Active Test Projects



Panes

SAS Event Stream Processing Studio contains *panes* that enable you to view different types of information within the same page. The following figure displays the **Engine Definitions** page, which contains a bottom pane with two tiles: **Associated Projects** and **Identification**.

Example of a Page with a Pane



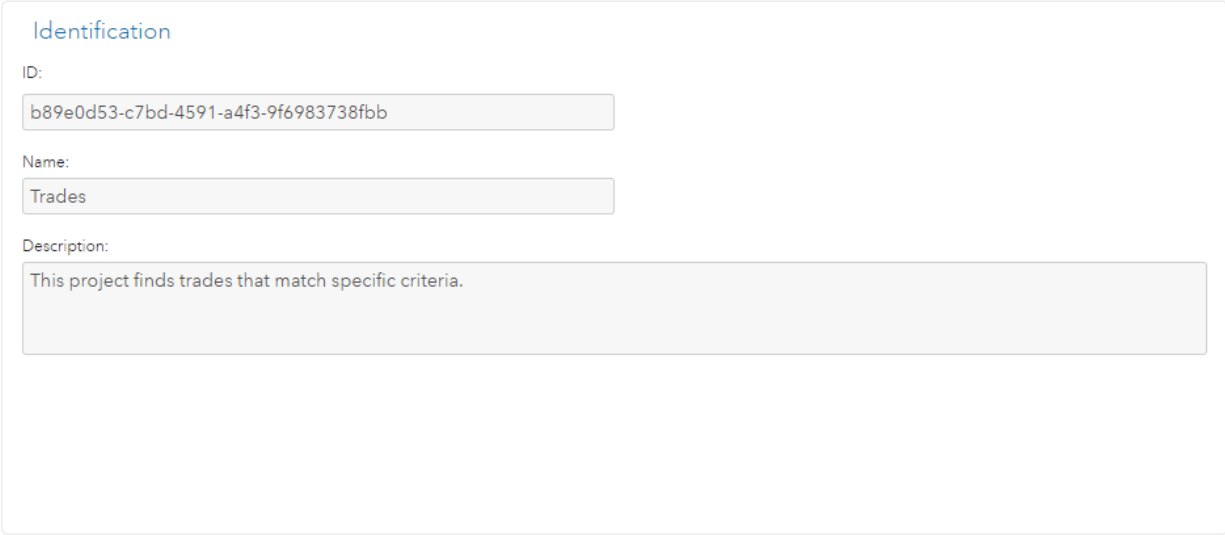
To resize a pane, drag a border that is marked with ····· in the appropriate direction. To resize a horizontal pane, drag the border upward or downward. To resize a vertical pane, drag the border left or right.

To hide a pane, click  . To display it again, click  .

Tiles

A *tile* is a self-contained block of information that resides within a pane or sometimes directly on a page. The same tile can appear on several pages. For example, the **Identification** tile appears on more than one page.

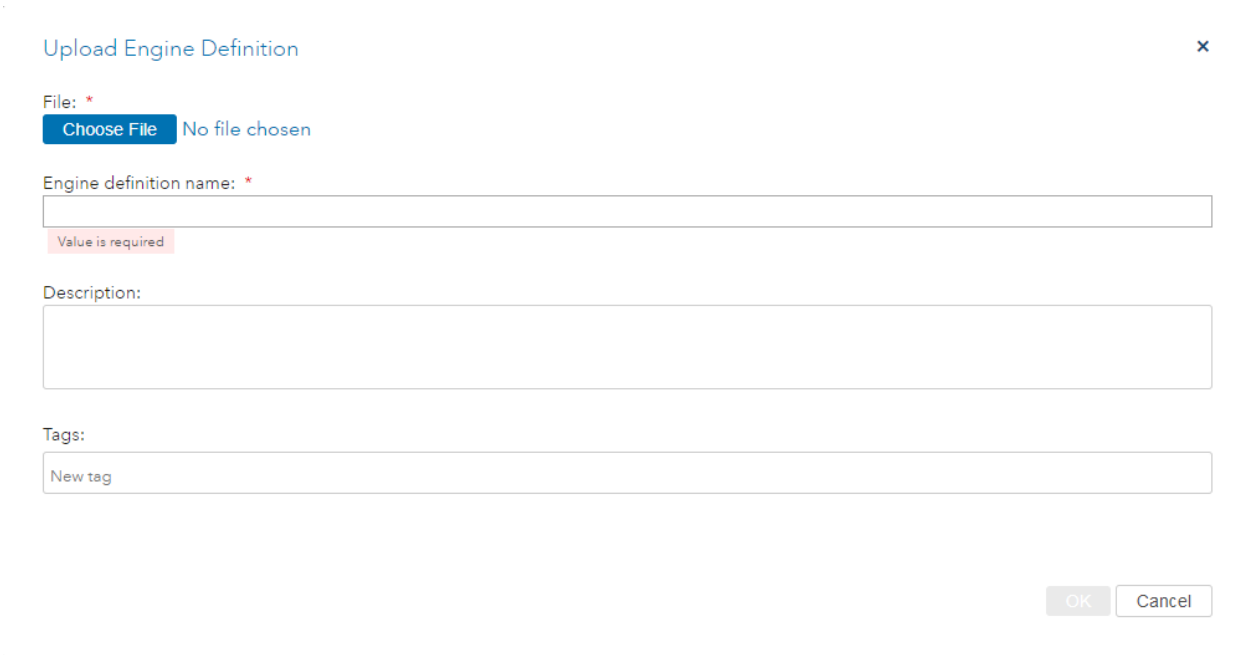
Example of a Tile



Windows

A *window* is a floating user interface element that often appears as a result of a user action. Windows generally provide a means by which to perform an action and can be closed to return you to the page from which the window was launched. The following figure shows a window that is used to upload an engine definition to SAS Event Stream Processing Studio.

The Upload Engine Definition Window

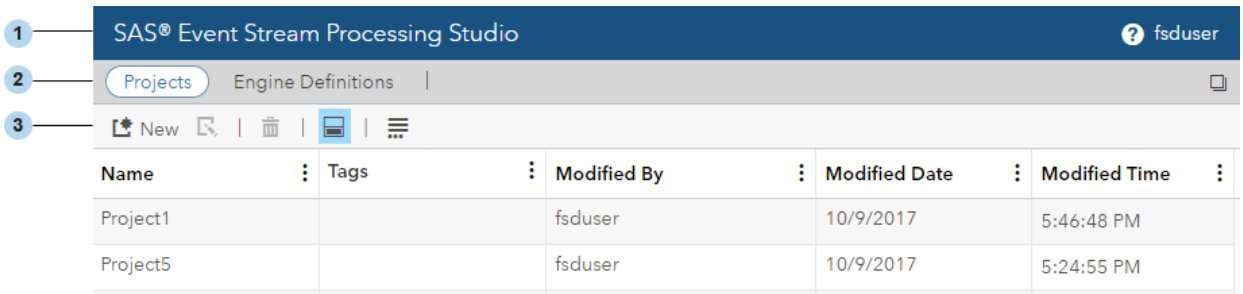


Note: The user interface element *window* in SAS Event Stream Processing Studio does not have the same meaning as a SAS Event Stream Processing window. In SAS Event Stream Processing, windows are components of a continuous query. A continuous query contains a source window and one or more derived windows. SAS Event Stream Processing windows are connected by edges, which have an associated direction.

SAS Event Stream Processing Studio contains both user interface element windows and SAS Event Stream Processing windows.

Toolbars

SAS Event Stream Processing Studio Toolbars



There are three main toolbars in SAS Event Stream Processing Studio, as shown in the following figure:

Item Number	Description	Name
1	Application bar	■ States whether you are currently viewing projects or engine definitions ■ Displays your user ID **Note:** If SAS Event Stream Processing Studio has been configured so that you do not need to log on with a user name and password, your user ID is not displayed. The screen shots in this section display a user ID. ■ Provides access to Help and product information ■ Enables you to sign out of SAS Event Stream Processing Studio
2	Menu bar	■ Provides access to the main SAS Event Stream Processing Studio pages: **Projects** and **Engine Definitions** ■ Provides access to each project, model test, or engine definition that is currently open
3	Toolbar	Includes buttons or tabs associated with the open item

Sorting and Filtering Items

To make it easier to work with a large amount of information, you can sort and filter items displayed in tables. You can also show, hide, and reorder columns.

You can sort lists of items by ascending or descending order. To sort in ascending order, click the heading of the column that you want to sort. To sort in descending order, click the column again. To remove sorting, click the column a third time.

You can create filter criteria by which to display only a subset of information for a column. To create filter criteria, click ⋮ for the column that you want to apply filter criteria to, select **Filter**, and enter your filter criteria.

You can configure the columns that you want to display. To do this, click ⋮ in any column, select **Columns**, and deselect the columns that you do not want to appear.

You can re-order columns. To do this, click and hold the column heading, and drag it to a different location.

SAS Event Stream Processing Studio Modeler

When you create a new project or open an existing project, a separate page that contains the project content appears. This project page displays SAS Event Stream Processing Studio Modeler, which enables you to design models in a visual way and to test them. SAS Event Stream Processing Studio Modeler also includes the XML Editor, which you can use as an alternative way to construct your model.

For more information about the modeler, see "Using SAS Event Stream Processing Studio Modeler" on page 14. For more information about the XML Editor, see "Using the XML Editor" on page 18.

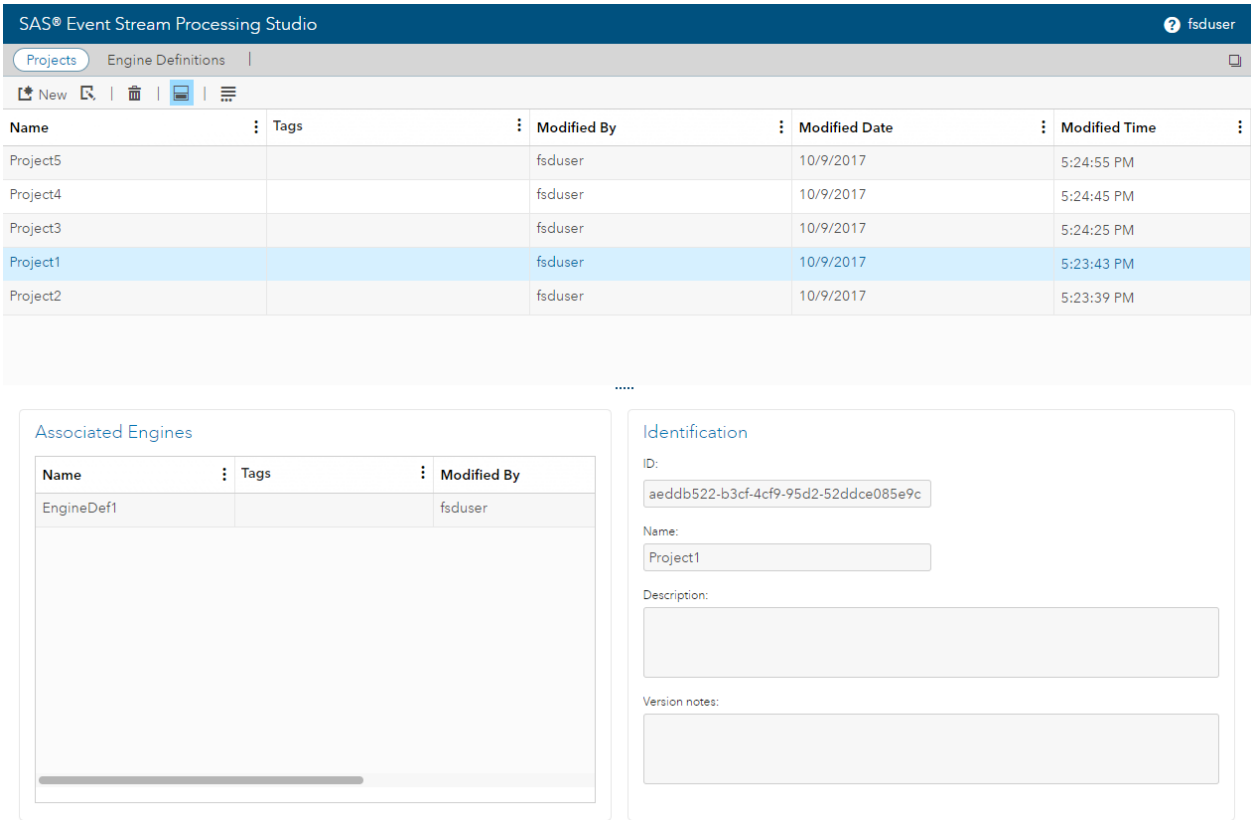
Working with Projects in SAS Event Stream Processing Studio

A *project* consists of one or more continuous queries. You can use SAS Event Stream Processing Studio to create, upload, download, and delete SAS Event Stream Processing projects. You can associate your project with an engine defined in SAS Event Stream Processing Studio. For more information, see "Creating Engine Definitions" on page 12.

Overview

The **Projects** page in SAS Event Stream Processing Studio enables you to view the projects in your deployment, along with their identification details and associated engines.

The Projects Page



Note: You can access the Manage Test Servers window from the **Projects** page, enabling you to register additional test servers or to display the details of existing test servers. To do this, click  and select **Manage test servers** from the drop-down list. For more information, see "Managing Test Servers in SAS Event Stream Processing Studio" on page 23.

Creating Projects

To create a new project:

1 On the **Projects** page, click  .

The New Project window appears.

2 In the New Project window, do the following:

a In the **Name** field, enter a name for the project.

b In the **Description** field, enter a description for the project.

c In the **Tags** field, enter any identifying keywords that describe the project.

d In the **Notes** field, enter any extra information relating to the project.

e Click **OK**.

If you do not currently have any test servers configured, you are prompted to decide whether you want to configure a test server now.

f Click **Yes** to configure a test server now or click **No** to configure a test server later.

3 If you chose to configure a test server now, do the following:

a In the **Name** field, enter a name to identify the new test server that you want to create.

b In the **Host** field, enter the host name or IP address of the server containing the new test server.

c In the **HTTP port** field, enter the new test server's administration port number.

d If required, in the **Tags** field, enter any identifying keywords that describe the new test server.

e If required, change the setting for the **Authentication** field:

- **None**: This is the default option.
- **Authenticate using an OAuth token**: This option is relevant only if the test server is configured to require authorization. If you select this option, an additional field appears where you must enter the OAuth token.
- **Authenticate using a user name and password**: This option is relevant only if the test server is configured to require authorization. If you select this option, additional fields appear where you must enter the user name and password.

f If required, select the **Connect using SSL** check box. Selecting this option is relevant only if the test server is configured to require SSL encryption.

g Click **OK**.

Note: To register additional test servers or to display the details of the test server that you have selected, in SAS Event Stream Processing Modeler, click **Manage Test Servers** in the **Test Server** drop-down list.

4 Click **OK**.

SAS Event Stream Processing Modeler appears.

Your project is created with a set of default properties. Before you start creating your model, it is recommended that you configure your project's properties.

To configure your new project's properties:

a Review the default project properties in the right pane and modify them if necessary.

b You can also add or modify additional project properties, such as SAS Micro Analytic Service modules, user-defined properties, and connector orchestration.

5 Click .

Note: To create a copy of the project with a different filename, click . Enter the relevant information into the Save As window and click **OK**.

Uploading Projects

To upload an existing project:

1 On the **Projects** page, click and select **Upload**.

The Upload Project window appears.

2 In the **File** field, click **Choose File**.

3 Navigate to the file containing the project that you want to upload and click **Open**.

4 In the **Name** field, rename the project if necessary.

5 In the **Description** field, enter a description for the project.

If the project that you are uploading contains a project description, the **Description** field is automatically populated with that text. The project description must be contained in a `description` element that is located directly within the `project` element.

6 In the **Tags** field, enter any identifying keywords that describe the project.

7 Click **OK**.

The uploaded project appears on the **Projects** page.

Downloading Projects

To download a project, select the project that you want to download from the table on the **Projects** page, click  , and select **Download**.

The project is downloaded to your computer.

Note: The location of the downloaded project might vary depending on your browser's configuration.

Deleting Projects

To delete a project, select the project that you want to delete from the table on the **Projects** page and click  . Click **OK** to confirm the deletion.

The project is permanently deleted from SAS Event Stream Processing Studio.

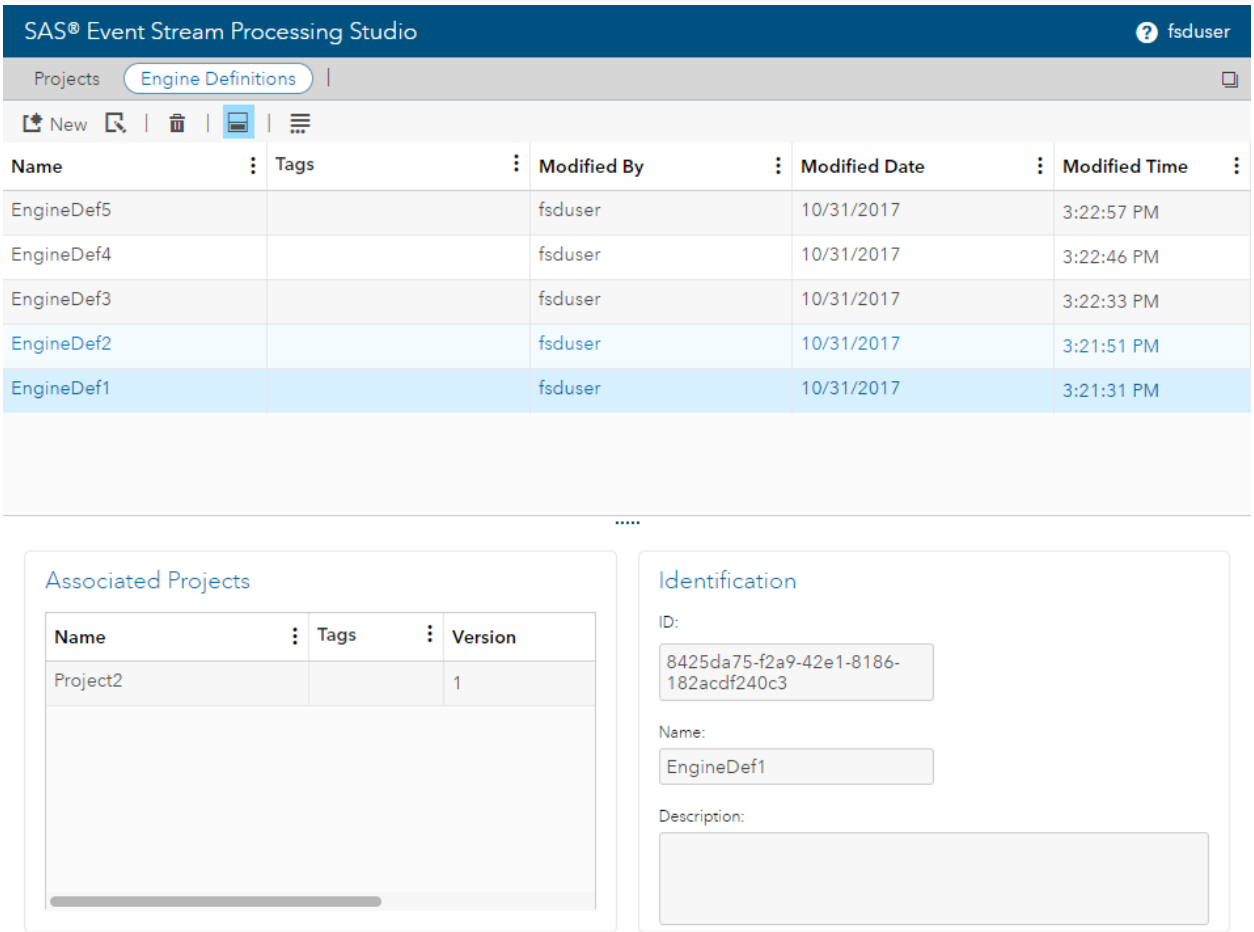
Working with Engine Definitions in SAS Event Stream Processing Studio

An *engine* is the top-level container in the SAS Event Stream Processing model hierarchy. Each model contains only one engine instance with a unique name. You can use SAS Event Stream Processing Studio to create, upload, download, and delete SAS Event Stream Processing engine definitions. You can associate each project that you produce or upload with an engine definition in SAS Event Stream Processing Studio.

Overview

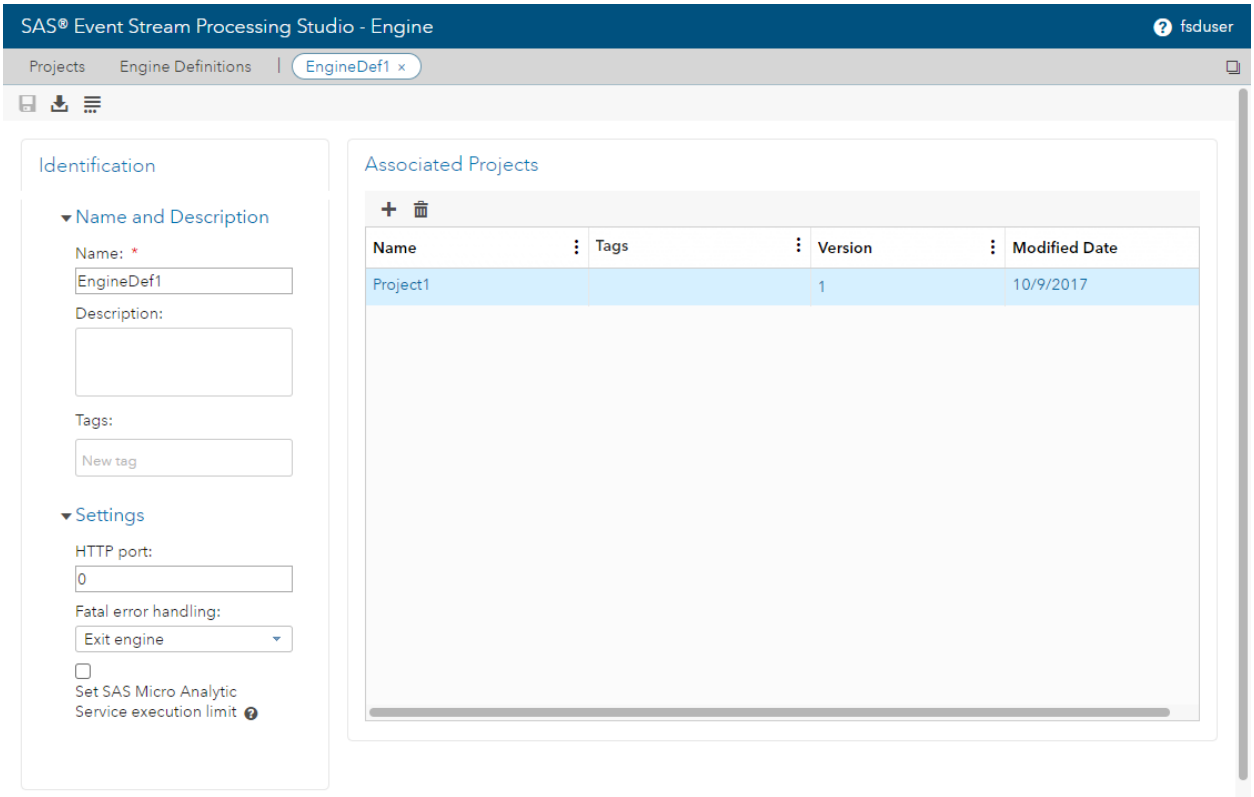
The **Engine Definitions** page in SAS Event Stream Processing Studio enables you to view all operational engines in your deployment, along with their identification details. You can view a list of the engine's associated projects in the **Associated Projects** tile.

The Engine Definitions Page



Double-clicking on an engine definition displays an engine definition page. This page consists of an **Identification** tile containing identifying information about the engine definition and an **Associated Projects** tile enabling you to associate the engine definition with one or more projects. Associating an engine definition with one or more projects is useful when executing multiple projects as a single action by grouping the projects within an engine. This enables you to reuse projects without having to re-create individual projects within each new engine.

An Engine Definition Page



Creating Engine Definitions

To create a new engine definition:

1 On the **Engine Definitions** page, click New.

The New Engine Definition window appears.

2 In the **Engine definition name** field, enter a name for the engine definition that you are creating.

3 Click **OK**.

Your engine definition is created, and an **Engine Definition** page appears.

4 Review the information in the **Identification** tile.

- the **Name** field contains the engine definition's name
- in the **Description** field, enter a description for the engine definition that you are creating
- in the **Tags** field, enter any keywords that describe the engine definition that you are creating
- update any other fields as required

5 Associate projects with your engine definition:

a In the **Associated Projects** pane, click +.

The Add Project window appears.

b In the Available projects table, select the project that you want your engine definition to be associated with and click .

The project that you selected appears in the Selected projects table.

c Click **Save**.

The newly associated project appears in the **Associated Projects** tile.

6 Click .

Uploading Engine Definitions

To upload an engine definition:

1 On the **Engine Definitions** page, click and select **Upload**.

The Upload Engine Definition window appears.

2 In the **File** field, click **Choose File**.

3 Navigate to the file containing the engine definition that you want to upload and click **Open**.

4 In the **Engine definition name** field, if necessary, adjust the name of the engine definition that you are uploading.

5 In the **Description** field, enter a description for the engine definition that you are uploading.

6 In the **Tags** field, enter any keywords that describe the engine definition that you are uploading.

7 Click **OK**.

Downloading Engine Definitions

To download an engine definition, select the engine definition that you want to download from the table on the **Engine Definitions** page, click , and select **Download**.

Note: The location of the downloaded engine definition might vary depending on your browser's configuration.

Deleting Engine Definitions

To delete an engine definition, select the engine definition that you want to delete from the table on the **Engine Definitions** page and click . Click **OK** to confirm the deletion.

The engine definition is deleted from SAS Event Stream Processing Studio.

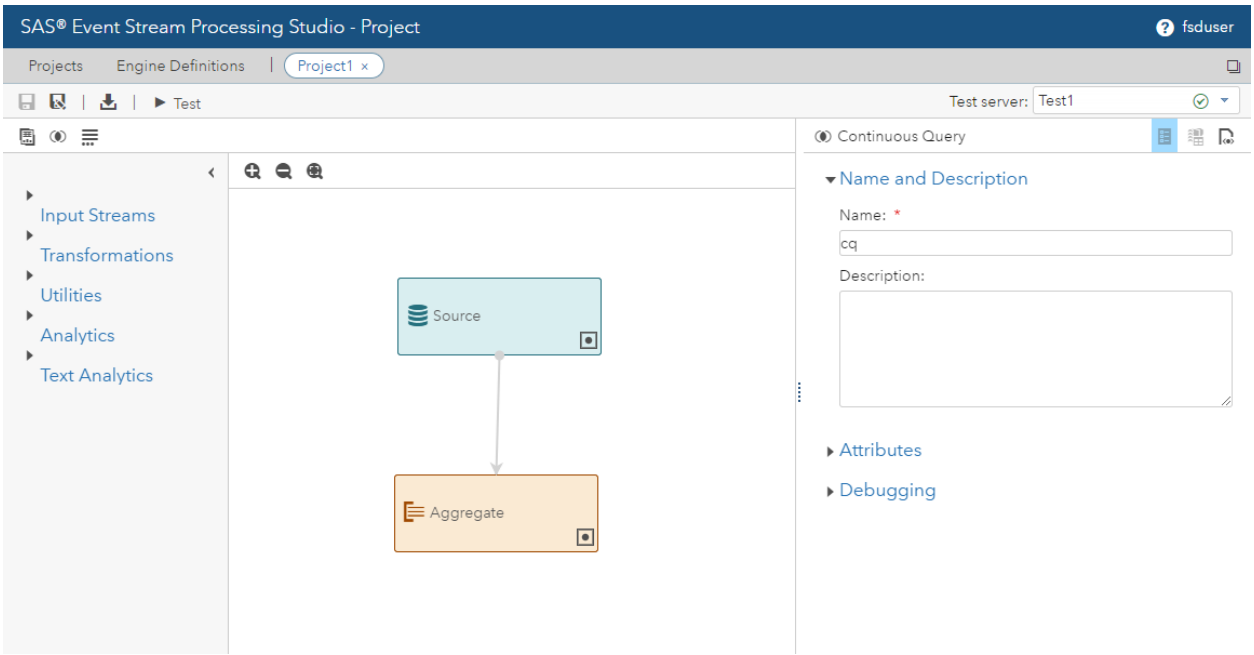
Using SAS Event Stream Processing Studio Modeler

Overview

SAS Event Stream Processing Studio Modeler enables you to construct, change, and test SAS Event Stream Processing models. A *model* specifies how a SAS Event Stream Processing engine analyzes and then transforms input event streams into meaningful results.

SAS Event Stream Processing Studio Modeler appears when you create a new project or open an existing project.

SAS Event Stream Processing Studio Modeler



Note: You can increase or decrease the magnification of your model by using the zoom buttons. Click to increase the magnification and click to decrease the magnification. To adjust the magnification of your model so that the entire model appears in the workspace, click .

The modeler displays one continuous query at a time. When you create a new model, a continuous query named `cq1` is created by default. To construct your model, you must configure at least one continuous query. For more information about configuring continuous queries, see "Configuring Continuous Queries in SAS Event Stream Processing Studio" on page 21.

Note: To select multiple windows in your model, press Ctrl and then click and drag the cursor to select each window. To pan your model, click anywhere in the workspace and then drag the cursor in the appropriate direction.

Configuring a Model's Test Server

When you create a model, if you do not have any test servers configured, you are prompted to decide whether you want to configure a test server. If you decide to create a test server, the test server that you create becomes the model's default test server and appears in SAS Event Stream Processing Studio Modeler:

If you want to test your model in test mode, you must create a test server and assign it to your model. However, creating a test server is optional when viewing or editing a model. For more information about testing your model, see "Testing Models in SAS Event Stream Processing Studio" on page 22. If you have multiple test servers configured, you can change the test server associated with your model by selecting an alternative test server from the **Test server** drop-down list.

Note: Functionality will be limited if you open a model that does not have a test server assigned to it. For example, connector properties are unavailable to models that do not have test servers assigned to them.

To manage your test servers, select **Manage test servers** from the **Test server** drop-down list. For more information about managing test servers, see "Managing Test Servers in SAS Event Stream Processing Studio" on page 23.

Adding Windows

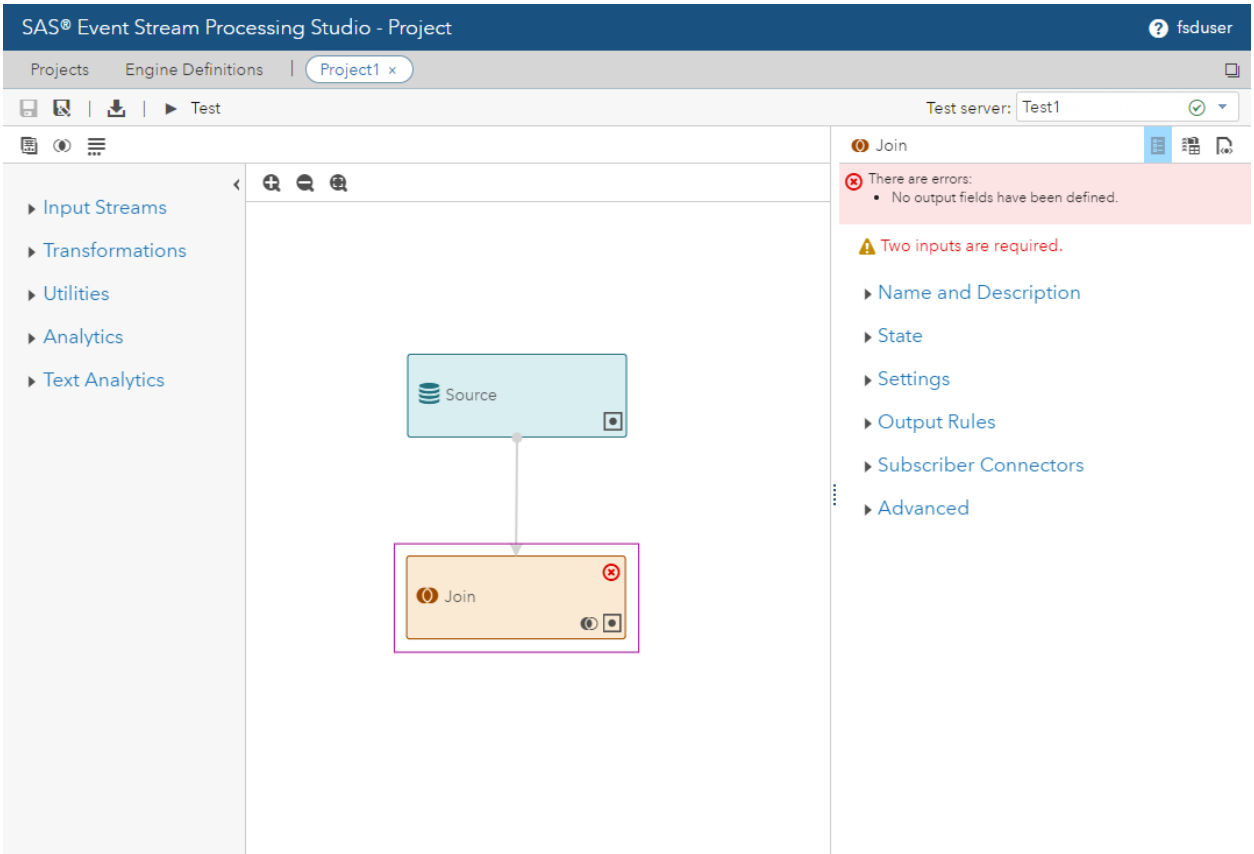
To add windows to the currently displayed continuous query, drag a window from the **Windows** pane on the left to the workspace.

The windows are grouped into the following categories:

- Input Streams
- Transformations
- Utilities
- Analytics — To use these windows, you must have a license for SAS Event Stream Processing Analytics. If you do not have a license, you can construct a model, but you cannot execute the model on an ESP server.
- Text Analytics — To use some of these windows, you must have a license for SAS Contextual Analysis (Text Category window and Text Context window) or SAS Sentiment Analysis (Text Sentiment window). If you do not have a license, you can construct a model, but you cannot execute the model on an ESP server.

You must ensure that you enter valid properties for each window. Entering invalid window properties causes the window to display a icon. In addition, the right pane displaying the window's properties displays a warning message. Here is an example of a model where the Join window contains invalid properties:

A Window Validation Warning Message

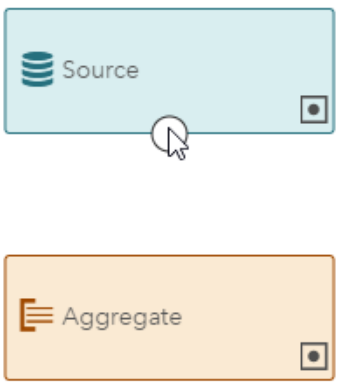


Connecting Windows

To connect a window to another window with an edge:

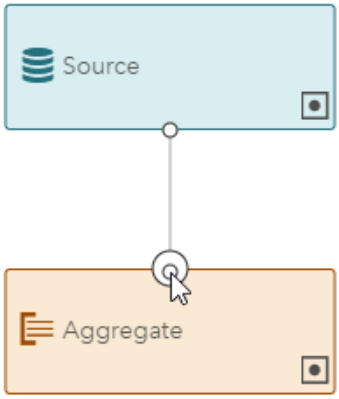
1 Position the cursor over the anchor point at the bottom of the window so that the anchor point changes from black to white:

Cursor over the Anchor Point



2 Click the white anchor point, hold the left mouse button down, and draw a line to the anchor point of another window:

An Edge Connecting Two Windows



The edge automatically connects to the window.

Note: You cannot change a connection by moving an edge from one window to another. Instead, you must establish a new connection by creating a new edge. If you have created a connection between two windows in error, you must delete the edge. To do this, select the edge that you want to delete and press Delete.

An invalid edge appears as a red dashed line in SAS Event Stream Processing Studio Modeler.

Deleting Windows and Edges

To remove a selected window or an edge from the model, press Delete.

Note: Deleting a window from a model automatically deletes all its connecting edges.

Window Icons

Each window in your model can display icons that represent its current state. For example, a Source window that contains a publisher connector displays . For information about each icon, see the following table:

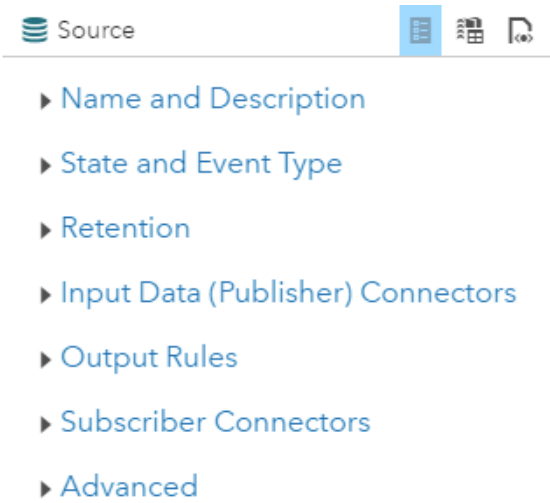
Icon	Description
	Indicates that the window's schema is invalid.
	Indicates that the window contains a fully stateful index. **Note:** If the window contains a non-stateful index, this icon is not displayed.
	Indicates that the Source window contains a connector. **Note:** Connector icons that appear on the left of a window indicate that the window contains a publisher connector. Connector icons that appear on the right of a window indicate that the window contains a subscriber connector.
	Indicates that the Join window contains an inner join type.

	Indicates that the Join window contains a full outer join type.
	Indicates that the Join window contains a right outer join type.
	Indicates that the Join window contains a left outer join type.

Configuring the Properties of Windows

To configure the properties of a window, click the window on the workspace. The right pane displays the properties for that window. Edit the properties as required.

Properties of the Source Window



Note: If the right pane displays XML code, you are viewing the XML Editor. Click in the right pane to display the properties instead.

Using the XML Editor

Overview

SAS Event Stream Processing Studio Modeler includes the XML Editor. You can use it as an alternative way of creating models, compared to the visual modeling capabilities in SAS Event Stream Processing Studio Modeler. The workspace displays a snapshot of your model's XML code. You can use the XML Editor to rename a window. To do this, select the window that you want to rename in the workspace and then change the window's name in the XML Editor.

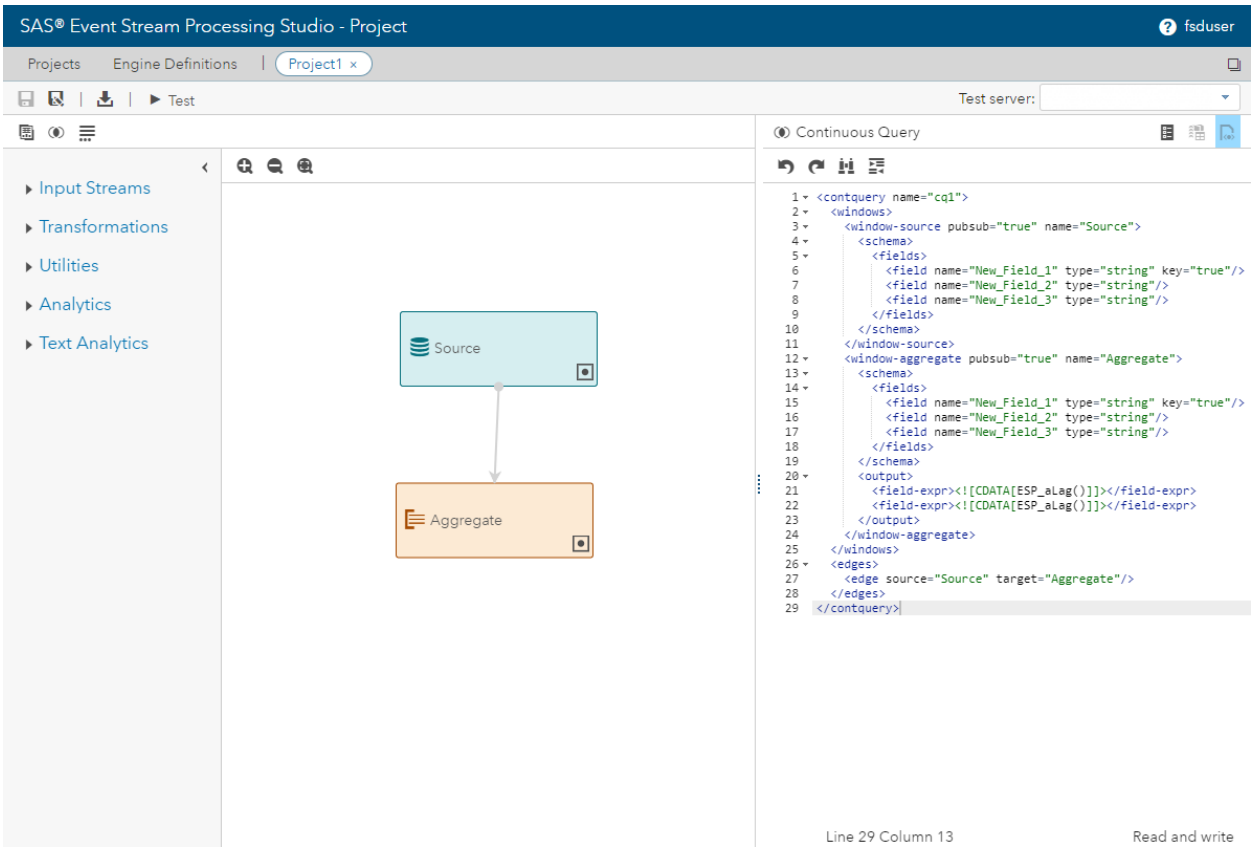
CAUTION! Manually editing your model's XML code using the XML Editor can result in an invalid model. Using SAS Event Stream Processing Studio Modeler to construct your model limits the possibility of your model containing invalid XML code. You must correct any invalid XML in the XML Editor before you can switch back to the **Properties** pane. Changes that you make manually in the XML Editor are not always reflected in the workspace.

Using the XML Editor to rename a window without first selecting it results in the window being replaced by a new window in a default position on the workspace. This invalidates your model. Any connections to or from the redundant window must then be deleted and then re-created in the workspace. Alternatively, you can manually edit the edges in the XML Editor.

To open the XML Editor, open a project and click  in the right pane.

The right pane displays the XML Editor.

The XML Editor within SAS Event Stream Processing Studio Modeler



Selecting a specific element in your workspace reloads the XML Editor to display only the corresponding section of XML code. To display the entire project's XML again, click  . If you have associated a project with an engine, the XML code that specifies the engine is not included in the project's XML code. This information is instead included as metadata in the SAS Event Stream Processing Studio database.

Clicking an area of white space in your workspace reloads the XML Editor to display the XML relating to your model's continuous query.

If you want to add a comment to your XML code, you must ensure that the comment is enclosed within its relevant XML element. If you do not, the comment will appear at the top of the XML Editor the next time the XML code is automatically reordered. The XML code is automatically reordered to maintain consistency with the SAS Event Stream Processing schema.

When a model is created, optional attributes are not included in its XML code. Models also contain settings that are not directly specified in the model's XML code, but they are represented in the user interface. For example, if a Source window has a default window state of **(inherit from query) pi_HASH**, the implied setting is not displayed in the XML code. See the example shown here:

The XML Editor Displaying a Source Window's XML Code without the Window's Default State

```
1 <window-source pubsub="true" name="Source1"/>
```

Changing the window's state from its default value includes the attribute in the model's XML code, as shown here:

The XML Editor Displaying a Source Window's XML Code with the Window's Default State

```
1 <window-source pubsub="true" index="pi_EMPTY" name="Source1"/>
```

Using Editing Tools and Keyboard Shortcuts

The XML Editor includes a toolbar that contains editing tools. These tools are also accessible using keyboard shortcuts.

Icon	Action	Keyboard Shortcut
	Reverts your previous change	Ctrl + Z
	Reverts the effects of the undo action	Ctrl + Y
	Searches for specific text	Ctrl + F
	Formats manually entered XML code	Not available
Not available	Removes the selected code from its original position	Ctrl + X
Not available	Copies the selected code to the clipboard	Ctrl + C
Not available	Pastes the code on the clipboard at the cursor's position	Ctrl + V

Validation

The XML Editor automatically validates the syntax of the code that you enter. If you enter invalid code, the XML Editor displays a warning:

Invalid XML Warning

Efficiency Tips

For some window types, you can use SAS Event Stream Processing Studio Modeler to copy schema fields if there are windows in your model that use the same fields. In the Output Schema window, click to open the Copy Fields from Input Schema window. Select the schema fields that you want to copy and click **OK**. Alternatively, you can use the XML Editor to copy and paste the fields between the windows.

Note: This functionality is not available for window types where it is not appropriate for schema fields to be copied from another window. For example, you cannot copy schema fields to or from windows that have implied or internally generated schemas.

Configuring Continuous Queries in SAS Event Stream Processing Studio

Continuous queries allow SAS Event Stream Processing engines to analyze and manipulate data. *Continuous queries* are queries that run automatically and periodically on data in real time.

SAS Event Stream Processing models must contain at least one continuous query. SAS Event Stream Processing Studio Modeler creates a continuous query `cq1` by default. You can then add and configure windows within this continuous query. Your model can contain many continuous queries.

Your continuous query must contain at least one Source window. Source windows connect to one or more derived windows (for example, a Pattern or Join window). After you have created a Source window, you can then add derived windows to your model.

To configure the properties of a continuous query:

1. On the **Projects** page, right-click the project that contains the continuous query that you want to configure, and select **Open Project**.

   SAS Event Stream Processing Studio Modeler appears. The right pane displays the project's properties.

2. Click on the toolbar.

   The right pane displays the properties of the continuous query that you selected when the project was last saved.

3. Configure the fields as required.

To add a new continuous query to your model, click on the toolbar and select **Add continuous query**. You can also delete continuous queries from your model by selecting **Delete continuous query**.

To switch between each continuous query in your model, select the continuous query that you want to view from the **Continuous Query** drop-down list on the toolbar.

The Continuous Query Drop-down List on the Toolbar

Testing Models in SAS Event Stream Processing Studio

You can use SAS Event Stream Processing Studio to verify that your model operates as intended. You can analyze how incoming data is transformed into meaningful resulting event streams consumed by subscribers.

Note: An *engine* is the top–level container in the SAS Event Stream Processing model hierarchy and can contain one or more projects. A project can contain one or more continuous queries. When you are running a model in test mode, only the project is tested. Test mode does not test engines.

1. On the **Projects** page, right-click the project containing the model that you want to test.
2. Select **Open project** from the menu.

   The project appears in a new page.
3. Click ▶ Test.

   A page appears, enabling you to test your model.
4. From the **Test server** drop-down list, select a test server to perform the test on.

   Note: If SAS Event Stream Processing Studio does not contain any registered test servers, you must register one before you can continue testing your model. To register a new test server, click +. To display the details of the test server that you have selected, click .
5. From the list of windows on the left of the screen, select the windows whose results you want to view.

   Note: The name of each continuous query is prepended to its corresponding window name with a period delimiter. For example, the ANPR window of the `geofence_cq` continuous query appears as geofence_cq.ANPR. If your model contains multiple continuous queries, the prepended continuous query enables you to easily differentiate between windows in each continuous query.
6. To run the test, click ▶.

   Each window's results appear in their corresponding tab. Only windows whose event stream you have subscribed to can display data. If you subscribed to view the results of six windows or fewer, you can choose to view your test results in windowed format by clicking .

   Alternatively, you can view your test results by opening the output file that you specified in your subscriber connector properties.

   You can use the **Display localized fields** check box to control whether information appear in the format of your browser's locale. The check box is selected by default. If you clear the check box, date and double data types appear in the format received from the ESP server instead. Regardless of whether the check box is selected, the time zone used to display the date information is always Greenwich Mean Time (GMT).

Note: This version of SAS Event Stream Processing Studio uses the WebSocket protocol to subscribe to windows. Therefore, models executed in this version of SAS Event Stream Processing Studio might display events in a different sequence than executing the model in a previous version of SAS Event Stream Processing Studio. The order of the events delivered to the WebSocket subscriber will not match the order of the events received from the engine. If you are using the WebSocket subscriber, the event key and event state take precedence over the sequence of events received from the engine.

7 To stop the test, click ■.

8 When you have finished testing your model, close the page.

Note: If the test fails and the error message does not provide enough information about what caused the failure, then SAS Event Stream Processing engine log files might provide additional information. For more information, see "Logging" in SAS Event Stream Processing: Troubleshooting.

Managing Test Servers in SAS Event Stream Processing Studio

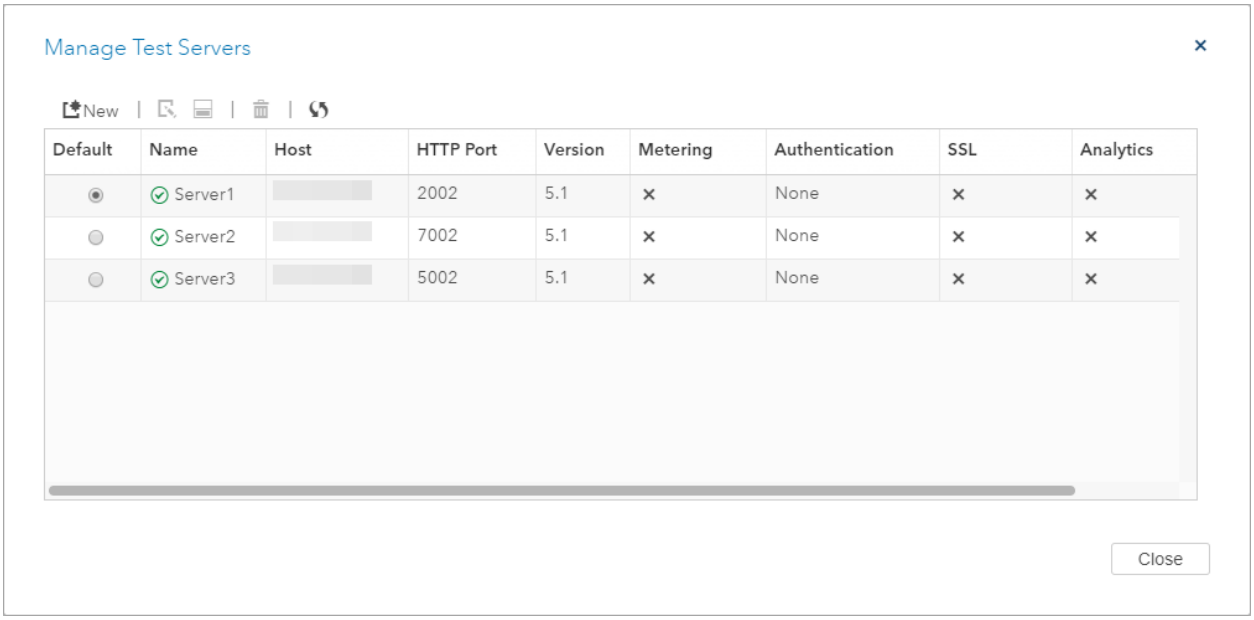
You can use the Manage Test Servers window to view, create, edit, and delete test servers in SAS Event Stream Processing Studio.

About Managing Test Servers

You can access the Manage Test Servers window from the **Projects** page. To do this, click and select **Manage test servers**. Alternatively, you can access the Manage Test Servers window from SAS Event Stream Processing Studio Modeler by selecting **Manage test servers** from the **Test server** drop-down list.

The following figure shows an example:

The Manage Test Servers Window, with Some Test Servers Listed



The Manage Test Servers window displays the following information for each test server:

- The default test server.
- The test server's health.
- The test server's name.
- The host on which the test server is running.
- The port used for HTTP administration requests.
- The SAS Event Stream Processing version installed on the host on which the test server is running.
- Whether SAS Event Stream Processing has been enabled to meter the number of events that are processed on the test server.
- Whether authentication is enabled on the test server. If authentication is enabled, the time at which it was enabled appears.
- Whether the SAS Event Stream Processing test server is configured to require SSL encryption.
- Whether a SAS Event Stream Processing Analytics license is installed on the host. To deploy a project that contains SAS Event Stream Processing Analytics windows, an appropriate license must be available.

The Name column displays an icon alongside the test server's name summarizing the test server's condition. The condition of the test server is determined by the server's connectivity and availability. This information helps you focus on the test servers that have problems. The following icons can appear in the Name column:

Icon	Description
Healthy	The test server is available and operating normally.
Errors Reported	The test server is not available.

You can also view additional information relating to each test server in your deployment, such as the connector types and analytics available.

To view a test server's additional information, select the test server whose additional information you want to view and click .

A tile appears containing additional information relating to the test server, as shown here:

Test Server Details

Server1

Publisher connector types:	adapter, bacnet, db, fs, opcua, project, url, uvc, websocket
Subscriber connector types:	adapter, db, fs, nurego, opcua, smtp, tdlistener
Calculate window analytics:	Correlation, DistributionFitting, SegmentedCorrelation, STFT, Summary, Tokenization, Tutorial, TextVectorization, Histogram, FitStat, ROC, ImageProcessing, MRR
Train window analytics:	DBSCAN, KMEANS, LinearRegression, SVM, LogisticRegression
Score window analytics:	DBSCAN, KMEANS, LinearRegression, SVM, LogisticRegression

To hide the additional information again, click .

Create a Test Server

To create a new test server:

1 Click New.

The Event Stream Processing Server window appears.

2 In the **Name** field, enter a name to identify the new test server.

3 In the **Host** field, enter the new test server's host name or IP address.

4 In the **HTTP port** field, enter the new test server's administration port number.

5 If required, in the **Tags** field, enter any identifying keywords that describe the new test server.

6 If required, change the setting for the **Authentication** field:

- **None**: This is the default option.
- **Authenticate using an OAuth token**: This option is relevant only if the ESP server is configured to require authorization. If you select this option, an additional field appears where you must enter the OAuth token.
- **Authenticate using a user name and password**: This option is relevant only if the ESP server is configured to require authorization. If you select this option, additional fields appear where you must enter the user name and password.

7 If required, select the **Connect using SSL** check box. Selecting this option is relevant only if the test server is configured to require SSL encryption.

8 Click **OK**.

Edit a Test Server

To edit an existing test server's details:

1 Select the test server that you want to open and click .

The Edit Event Stream Processing Server window appears.

2 Edit the fields as required.

3 Click **OK**.

Delete a Test Server

Deleting a test server removes it from the table on the Manage Test Servers window. Deleting test servers can be useful if the table contains test servers that are no longer used. You can delete a test server that is still running. To delete a specific test server from the table:

1 Select the test server that you want to delete and click .

2 Click **Yes**.

Refresh the Manage Test Servers Window

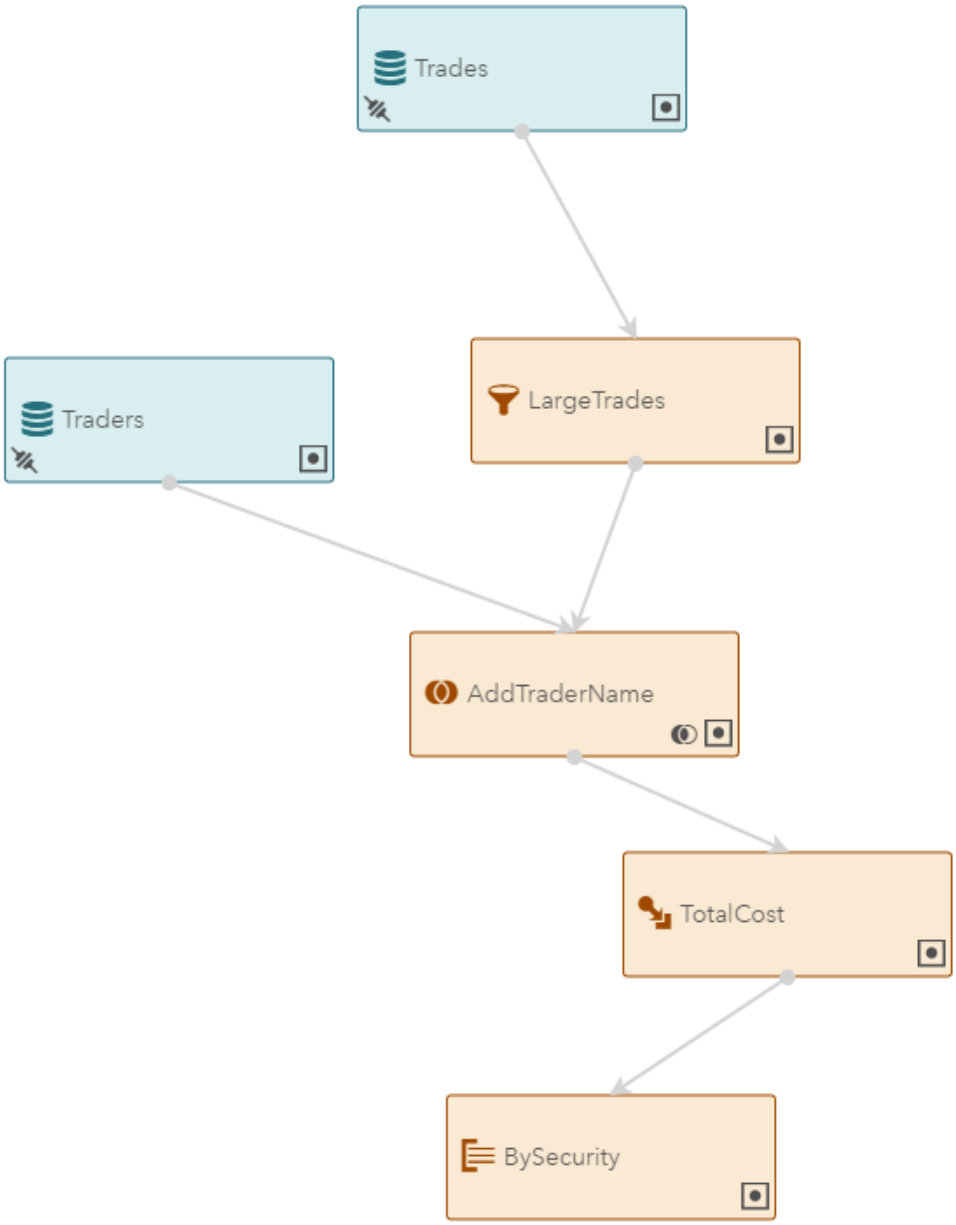
To refresh the table of test servers, click .

Example: Processing Trades

This example creates a model that processes stock market trades. The model identifies large securities transactions and the traders who were involved in those trades. The model performs the following actions:

- streams events about stock market securities transactions using a Source window called Trades
- receives information about traders using a Source window called Traders
- identifies large trades using a Filter window called LargeTrades
- matches the large trades with the traders who made those trades using a Join window called AddTraderName
- computes the total cost of the large trades using a Compute window called TotalCost
- aggregates the large trades by security using an Aggregate window called BySecurity

Diagram of the Trades Model



Note: The comma-separated value (CSV) data and model XML code used in this example are available within your SAS Event Stream Processing installation, typically in the following location: `/opt/sas/viya/home/SASEventStreamProcessingEngine/<release>/examples/xml/trades_xml`. Replace *<release>* with the release number in your SAS Event Stream Processing installation directory path.

1 On the **Projects** page, click New.

   The New Project window appears.

2 In the New Project window, do the following:

   a In the **Name** field, enter `Trades`.

b In the **Description** field, enter a description. Here is an example: `This model can be used to identify large securities transactions and the traders who were involved in those trades.`

c Click **OK**.

If you do not currently have any test servers configured, you are prompted to decide whether you want to configure a test server now.

Note: It is assumed that you do not have any test servers configured. If you already have test servers configured, go to step 5.

3 Click **Yes** to configure a test server now.

The Event Stream Processing Server window appears.

4 Configure a test server:

a In the **Name** field, enter a name to identify the new test server that you want to create.

b In the **Host** field, enter the host name or IP address of the new test server.

c In the **HTTP port** field, enter the new test server's HTTP publish/subscribe port.

d If required, in the **Tags** field, enter any identifying keywords that describe the new test server.

e If required, change the setting for the **Authentication** field:

- **None**: This is the default option.
- **Authenticate using an OAuth token**: This option is relevant only if the ESP server is configured to require authorization. If you select this option, an additional field appears where you must enter the OAuth token.
- **Authenticate using a user name and password**: This option is relevant only if the ESP server is configured to require authorization. If you select this option, additional fields appear where you must enter the user name and password.

f If required, select the **Connect using SSL** check box. Selecting this option is relevant only if the test server is configured to require SSL encryption.

g Click **OK**.

Your project is created with a set of default properties.

5 In the right pane, configure your project's properties:

a In the **Name** field, change the default name to `trades_proj`.

b Expand **Attributes**.

c In the **Threads** field, change the thread pool size to `4`.

6 Configure the project's continuous query:

a Click  .

b In the right pane, in the **Name** field, change the continuous query's default name `cq1` to `trades_cq`.

7 Expand **Input Streams** on the **Windows** pane on the left and drag a Source window to the workspace.

The right pane displays the Source window's properties.

This window receives events about stock market securities transactions.

8 Specify a name and description for the Source window:

a In the right pane, in the **Name** field, change the default name to `Trades`.

b In the **Description** field, enter `Trades Source window`.

9 Specify an output schema for the Trades window:

a In the right pane, click .

b Click .

The Output Schema window appears.

c Click to add a row to the schema table. After you add a row, click again to add the next row.

Enter the following values in the rows:

Key	Field Name	Type
Y	tradeID	String
N	security	String
N	quantity	Int32
N	price	Double
N	traderID	Int64
N	time	Timestamp

d Click **OK**.

10 Configure the Trades window to stream events from a file called trades.csv that contains stock market securities transactions. You can find this example CSV file in the `trades_xml` folder in the `examples` directory. To add a connector to this CSV file:

a In the right pane, click .

b Expand **Input Data (Publisher) Connectors**.

c Click .

The Publisher Connectors window appears.

d In the **Name** field, replace the default value with `TradesConnector`.

e In the **Fsname** field, enter the path to the CSV file. For example, you might enter `/opt/sas/viya/home/SASEventStreamProcessingEngine/<release>/examples/xml/trades_xml/trades.csv`. Replace *<release>* with the release number in your SAS Event Stream Processing installation directory path.

f In the **Fstype** drop-down list, select **csv**.

g Configure the TradesConnector connector's properties:

i Click **All properties**.

The All Properties window appears.

ii Enter `%d/%b/%Y:%H:%M:%S` in the **dateformat** property's **Value** field.

iii Click **OK**.

h Click **OK**.

11 Collapse **Input Data (Publisher) Connectors**.

12 Specify a state for the Trades window:

a Expand **State and Event Type**.

b In the **Window state and index** drop-down list, select **Stateful (pi_RBTREE)**.

13 Expand **Input Streams** on the **Windows** pane on the left and drag another Source window to the workspace.

The right pane displays the Source window's properties.

Configure this window to receive information about stock market traders.

14 Specify a name and description for the Source window:

a In the right pane, in the **Name** field, change the default name to `Traders`.

b In the **Description** field, enter `Traders Source window`.

15 Specify an output schema for the Traders window:

a In the right pane, click .

b Click .

The Output Schema window appears.

c Click to add a row to the schema table. After you add a row, click again to add the next row.

Enter the following values:

Key	Name	Type
Y	ID	Int64
N	name	String

d Click **OK**.

16 Configure the Traders window to receive information from a file called traders.csv that contains details of stock market traders. You can find this example CSV file in the `trades_xml` folder in the `examples` directory. To add a connector to this CSV file:

a In the right pane, click .

b Expand **Input Data (Publisher) Connectors**.

c Click .

The Publisher Connectors window appears.

d In the **Name** field, replace the default value with `TradersConnector`.

e In the **Fsname** field, enter the path to the CSV file. For example, you might enter **`/opt/sas/viya/home/SASEventStreamProcessingEngine/<release>/examples/xml/trades_xml/traders.csv`**. Replace *<release>* with the release number in your SAS Event Stream Processing installation directory path.

f In the **Fstype** drop-down list, select **csv**.

g Click **OK**.

17 Expand **Transformations** on the **Windows** pane on the left and drag a Filter window to the workspace.

Configure this window to identify large trades. In this example, a trade is regarded as a large trade if the quantity of stock traded equals or exceeds 100.

18 Click the newly created Filter window on the workspace.

The right pane displays the Filter window's properties.

19 Specify a name and description for the Filter window:

a In the **Name** field, change the default name to **`LargeTrades`**.

b In the **Description** field, enter **`LargeTrades Filter window`**.

20 Specify a filter expression for the LargeTrades window:

a Expand **Filter**.

b In the **Expression** field, enter `quantity >= 100`

c Collapse **Filter**.

21 Connect the Trades window to the LargeTrades window with an edge:

a Position the cursor over the anchor point at the bottom of the Trades window so that the anchor point changes from black to white.

b Click the white anchor point, hold the mouse button down, and draw a line to the anchor point in the LargeTrades window.

The LargeTrades window now accepts trades from the Trades window.

22 Expand **Transformations** on the **Windows** pane on the left and drag a Join window to the workspace.

Configure this window to match large trades with the traders who made those trades.

23 Specify a name and description for the Join window:

a In the right pane, in the **Name** field, change the default name to **`AddTraderName`**.

b In the **Description** field, enter **`AddTraderName Join window`**.

24 Connect the LargeTrades window to the AddTraderName window with an edge.

The AddTraderName window now accepts trades from the LargeTrades window.

25 Connect the Traders window to the AddTraderName window with an edge.

The AddTraderName window now accepts trader names from the Traders window.

26 Click the AddTraderName window on the workspace.

The right pane displays the AddTraderName window's properties.

27 In the right pane, expand **Settings** and notice that the LargeTrades window is regarded as the left window and the Traders window is regarded as the right window. This is due to the order in which you added the edges.

28 Configure the AddTraderName window's join conditions:

a Expand **Join Conditions**.

b In the **Join Conditions** section, click to add a row to the table.

c Click the cell in the Left: LargeTrades column, and select **traderID**.

d Click the cell in the Right: Traders: column, and select **ID**.

e Collapse **Join Conditions**.

29 Configure the AddTraderName window's join criteria:

a Confirm that the **Join Type** drop-down list has a default value of **LeftOuter**.

b Collapse **Join Criteria**.

30 Specify a schema for the AddTraderName window:

a In the right pane, click .

b Click .

The Output Schema window appears. Use this window to configure the fields as shown in the following table. The schema fields required have already been defined previously. Click to open the Copy Fields from Input Schema window. Select the following schema fields and click **OK**.

Window	Field	Type
LargeTrades	security	String
LargeTrades	quantity	Int32
LargeTrades	price	Double
LargeTrades	traderID	Int64
LargeTrades	time	Timestamp
Traders	name	String

The Output Schema window displays the fields that you selected.

c Click **OK**.

31 Click .

32 Expand **Transformations** on the **Windows** pane on the left and drag a Compute window to the workspace.

The right pane displays the Compute window's properties.

Configure this window to compute the total cost of the large trades.

33 Specify a name and description for the Compute window:

a In the **Name** field, change the default name to `TotalCost`.

b In the **Description** field, enter `TotalCost Compute window`.

34 Connect the AddTraderName window to the TotalCost window with an edge.

The TotalCost window now accepts trades from the AddTraderName window.

35 Click the TotalCost window to display the Compute window's properties in the right pane again.

36 Specify an output schema for the TotalCost window:

a In the right pane, click .

b Click .

The Output Schema window appears.

c Click to add a row to the schema table.

Enter the following values:

Note: Alternatively, you can copy schema fields that you have previously defined. Click to open the Copy Fields from Input Schema window. Select the schema fields that you want to copy and click **OK**.

Key	Field Name	Type	Expression
Y	tradeID	String	(not applicable)
N	security	String	security
N	quantity	Int32	quantity
N	price	Double	price
N	totalCost	Double	price*quantity
N	traderID	Int64	traderID
N	time	Timestamp	time
N	name	String	name

d Click **OK**.

37 Click .

38 Expand **Transformations** on the **Windows** pane on the left and drag an Aggregate window to the workspace.

The right pane displays the Aggregate window's properties.

Configure this window to compute the total cost of the large trades.

39 Specify a name and description for the Aggregate window:

a In the **Name** field, change the default name to `BySecurity`.

b In the **Description** field, enter `This window computes the total cost of the large trades`.

40 Connect the TotalCost window to the BySecurity window with an edge.

The BySecurity window now accepts trades from the TotalCost window.

41 Click the BySecurity window to display the Aggregate window's properties in the right pane again.

42 Specify an output schema for the BySecurity window:

a In the right pane, click .

b Click .

The Output Schema window appears.

c Click to add a row to the schema table. Enter the following values:

Key	Field Name	Type	Aggregate function	Parameters
Y	security	String	(not applicable)	(not applicable)
N	quantityTotal	Double	ESP_aSum	quantity
N	costTotal	Double	ESP_aSum	totalCost

d Click **OK**.

43 The model is now complete. Click to save your model.

44 Click ▶ Test.

A new page called **Test: trades_proj** appears.

45 In the **Test Server** drop-down list, select the test server on which you want to test the model.

46 Click ▶.

The results for each window appear on separate tabs:

- the **trades_cq.Trades** tab lists the stock market securities transactions
- the **trades_cq.Traders** tab lists the traders
- the **trades_cq.LargeTrades** tab lists the large trades
- the **trades_cq.AddTraderName** tab lists the large trades and includes an additional column that shows trader names
- the **trades_cq.TotalCost** tab includes an additional column that shows the total cost of each transaction. You can use this information to identify high-value transactions.
- the **trades_cq.BySecurity** tab shows all the inserts, deletes, and update blocks for the large trades, as illustrated in the following figure. The newest event is shown at the top of the table. The total cost of transactions for each security is displayed: 601300 for IBM and 91950 for SAP.

Note: If the table is empty, check that the publisher connectors for the Trades and Traders windows are set correctly to point to the CSV files.

47 To stop the test, click ■.

The project stops and then unloads from the SAS Event Stream Processing server.

Example: Streaming Analytics with Scoring and Training

This example demonstrates the use of the machine learning algorithm *k-means*, which is often used for cluster analysis in data mining. *K-means* clustering partitions observations into clusters with the nearest mean.

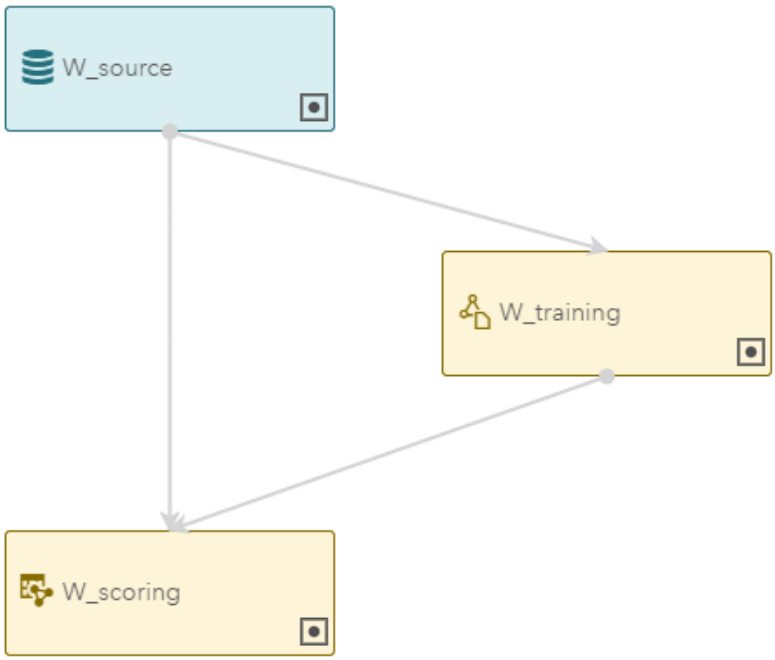
Note: To successfully complete this example, you must have access to SAS Event Stream Processing Analytics — a separately orderable and licensed package that enables the use of advanced analytics and machine learning techniques. Your deployment must also contain the esp_sa_plugin library.

The algorithm assigns data points to their nearest cluster centroid and recomputes each cluster centroid based on the average of data points belonging to the cluster.

The model contains a Source window that receives data to be scored, a Train window that generates and periodically updates the *k-means* model, and a Score window that performs the scoring. In *k-means* clustering, the input event is augmented with a cluster number indicating the cluster that the observation falls into.

Note: The CSV data and model XML code used in this example are available within your SAS Event Stream Processing installation, typically in the following location: **`/opt/sas/viya/home/SASEventStreamProcessingEngine/<release>/examples/xml/analytics_kmeans`**. Replace *<release>* with the release number in your SAS Event Stream Processing installation directory path.

Diagram of the Streaming Analytics Model with Scoring and Training



1 On the **Projects** page, click New.

The New Project window appears.

2 In the New Project window, do the following:

   a In the **Name** field, enter `Scoring_and_Training`.

   b In the **Description** field, enter: `This model demonstrates the use of the K-means machine learning algorithm for clustering.`

   c Click **OK**.

   If you do not currently have any test servers configured, you are prompted to decide whether you want to configure a test server now.

   Note: It is assumed that you do not have any test servers configured. If you already have test servers configured, go to step 5.

3 Click **Yes** to configure a test server now.

   The Event Stream Processing Server window appears.

4 Configure a test server:

   a In the **Name** field, enter a name to identify the new test server that you want to create.

   b In the **Host** field, enter the host name of the new test server.

   c In the **HTTP port** field, enter the new test server's HTTP publish/subscribe port.

   d If required, in the **Tags** field, enter any identifying keywords that describe the new test server.

   e If required, change the setting for the **Authentication** field:

      - **None**: This is the default option.
      - **Authenticate using an OAuth token**: This option is relevant only if the ESP server is configured to require authorization. If you select this option, an additional field appears where you must enter the OAuth token.
      - **Authenticate using a user name and password**: This option is relevant only if the ESP server is configured to require authorization. If you select this option, additional fields appear where you must enter the user name and password.

   f If required, select the **Connect using SSL** check box. Selecting this option is relevant only if the test server is configured to require SSL encryption.

   g Click **OK**.

   SAS Event Stream Processing Modeler appears.

5 In the right pane, configure your project's properties:

   a Expand **Attributes**.

   b Select the **Compress open patterns** check box.

6 Expand **Input Streams** on the **Windows** pane on the left and drag a Source window to the workspace.

   The right pane displays the Source window's properties.

7 Enter a name and description for the Source window:

   a In the right pane, in the **Name** field, change the default name to `W_source`.

   b In the **Description** field, enter `Source window`.

8 Specify an output schema for the W_source window:

a In the right pane, click .

b Click .

The Output Schema window appears.

c Click to add a row to the schema table. After you add a row, click again to add the next row.

Enter the following values in the rows:

Key	Field Name	Type
Y	id	Int64
N	x_c	Double
N	y_c	Double

d Click **OK**.

9 The W_source window will stream events from a file called input.csv that contains example data. You can find this CSV file in the `analytics_kmeans` folder in the `examples` directory. To add a connector to this CSV file:

a In the right pane, click .

b Click the W_source window.

c Expand **Input Data (Publisher) Connectors** and click .

The Publisher Connectors window appears.

d In the **Name** field, replace the default value with `Source_File`.

e In the **Fsname** field, enter the path to the CSV file. For example, you might enter `/opt/sas/viya/home/SASEventStreamProcessingEngine/<release>/examples/xml/analytics_kmeans/input.csv`. Replace *<release>* with the release number in your SAS Event Stream Processing installation directory path.

f In the **Fstype** drop-down list, select **csv**.

g Configure the Source_File connector's properties:

i Click **All properties...**.

The All Properties window appears.

ii Select **true** from the drop-down list in the **Value** field of the **transactional** property.

iii Enter `1` in the **Value** field of the **blocksize** property.

iv Click **OK**.

h Click **OK**.

i Collapse **Input Data (Publisher) Connectors**.

10 Configure an output rule for the W_source window:

a Expand **Output Rules**.

b Select the **Only output "insert" events** check box.

11 Expand **Analytics** on the **Windows** pane on the left and drag a Train window to the workspace.

This window will periodically generate a new clustering model using the *k-means* algorithm.

The right pane displays the Train window's properties.

12 Specify a name and description for the Train window:

a In the right pane, in the **Name** field, change the default name to `W_training`.

b In the **Description** field, enter `W_training window`.

13 Connect the W_source window to the W_training window with an edge:

a Position the cursor over the anchor point at the bottom of the W_source window so that the anchor point changes from black to white.

b Click the white anchor point, hold the mouse button down, and draw a line to the anchor point in the W_training window.

The W_training window now accepts events from the W_source window.

14 Click the W_training window on the workspace.

15 Expand **Settings**.

16 In the **Algorithm** drop-down list, select **KMEANS**.

17 Expand **KMEANS**:

a Expand **Parameters**.

b In the **nClusters** field, confirm that the default number of clusters is set to `2`.

c In the **initSeed** field, enter `1` to specify the random seed used during initialization when each point is assigned to a random cluster.

d In the **dampingFactor** field, confirm that the default value for damping factor for old data points is set to `0.8`.

e In the **fadeOutFactor** field, confirm that the default value for determining whether an existing cluster is fading out is set to `0.05`.

Note: If a cluster weight is smaller than the maximal cluster weight among other clusters multiplied by θ, then this cluster is considered to be fading.

f In the **disturbFactor** field, confirm that the default value for the disturbance factor when splitting a cluster is set to `0.01`.

g In the **nInit** field, confirm that the default value for the number of data events used during initialization is set to `50`.

h In the **velocity** field, enter `5` to specify the number of events arriving at a single timestamp.

i In the **commitInterval** field, confirm that the default value for the number of timestamps to elapse before committing a model to downstream scoring is set to `25`.

j Collapse **Parameters**.

k Expand **Input Map**.

l Click the **inputs** field twice to display the drop-down list and select **x_c** and **y_c** from the drop-down list to specify the variable names to use in clustering.

18 Expand **Analytics** on the **Windows** pane on the left and drag a Score window to the workspace.

The right pane displays the Score window's properties.

This window scores incoming events.

19 In the right pane, in the **Name** field, change the default name to `W_scoring`:

a In the right pane, in the **Name** field, change the default name to `W_scoring`.

b In the **Description** field, enter `Score window`.

20 Specify a schema for the W_scoring window:

a In the right pane, click .

b Click .

The Output Schema window appears.

c Click to add a row to the schema table. After you add a row, click again to add the next row.

Enter the following values in the rows:

Key	Field Name	Type
Y	id	Int64
N	x_c	Double
N	y_c	Double
N	seg	Int32
N	min_dist	Double
N	model_id	Int64

21 Click **OK**.

22 Connect the W_source window to the W_scoring window with an edge.

The W_scoring window can now score events that originate from the W_source window.

23 Configure the settings for the W_scoring window:

a Click the W_scoring window on the workspace.

b Click .

c Expand **Settings**.

d Specify an algorithm to use to score incoming events:

i In the **Configured algorithms** field, click .

The Configured Algorithms window appears.

ii Select the **KMEANS** check box.

iii Click **OK**.

e Expand **KMEANS**.

f Configure an input map:

i Expand **Input Map**.

ii Click the **inputs** field twice to display the drop-down list and select **x_c** and **y_c** from the drop-down list to specify the variable names to use in clustering.

iii Collapse **Input Map**.

g Configure an output map:

i Expand **Output Map**.

ii Specify the output variable name in the output schema that stores the cluster label. In the **labelOut** row, click the **Name** field twice to display the drop-down list and select **seg**.

iii Specify the output variable name in the output schema that stores the distance to the nearest cluster. In the **minDistanceOut** row, click the **Name** field twice to display the drop-down list and select **min_dist**.

iv Specify the output variable name in the output schema that stores the ID of the model from which the score is computed. In the **modelIdOut** row, click the **Name** field twice to display the drop-down list and select **model_id**.

24 Connect the W_training window to the W_scoring window with an edge.

25 Configure the project's continuous query:

a Click .

b In the right pane, in the **Name** field, change the default name to `scoretrain_cq`.

c Expand **Debugging**.

d In the **Trace in server log** field, select **W_scoring** and **W_training**.

26 Click .

27 Click ▸ Test.

A new page called **Test: Scoring_and_Training** appears.

28 In the **Test Server** drop-down list, select the test server on which you want to test the model.

29 Click ▶.

The results for each window appear in separate tabs:

- the **scoretrain_cq.w_source** tab displays events to be scored
- the **scoretrain_cq.w_training** tab displays the generated clustering model using the *k-means* algorithm
- the **scoretrain_cq.w_scoring** tab displays the scored events

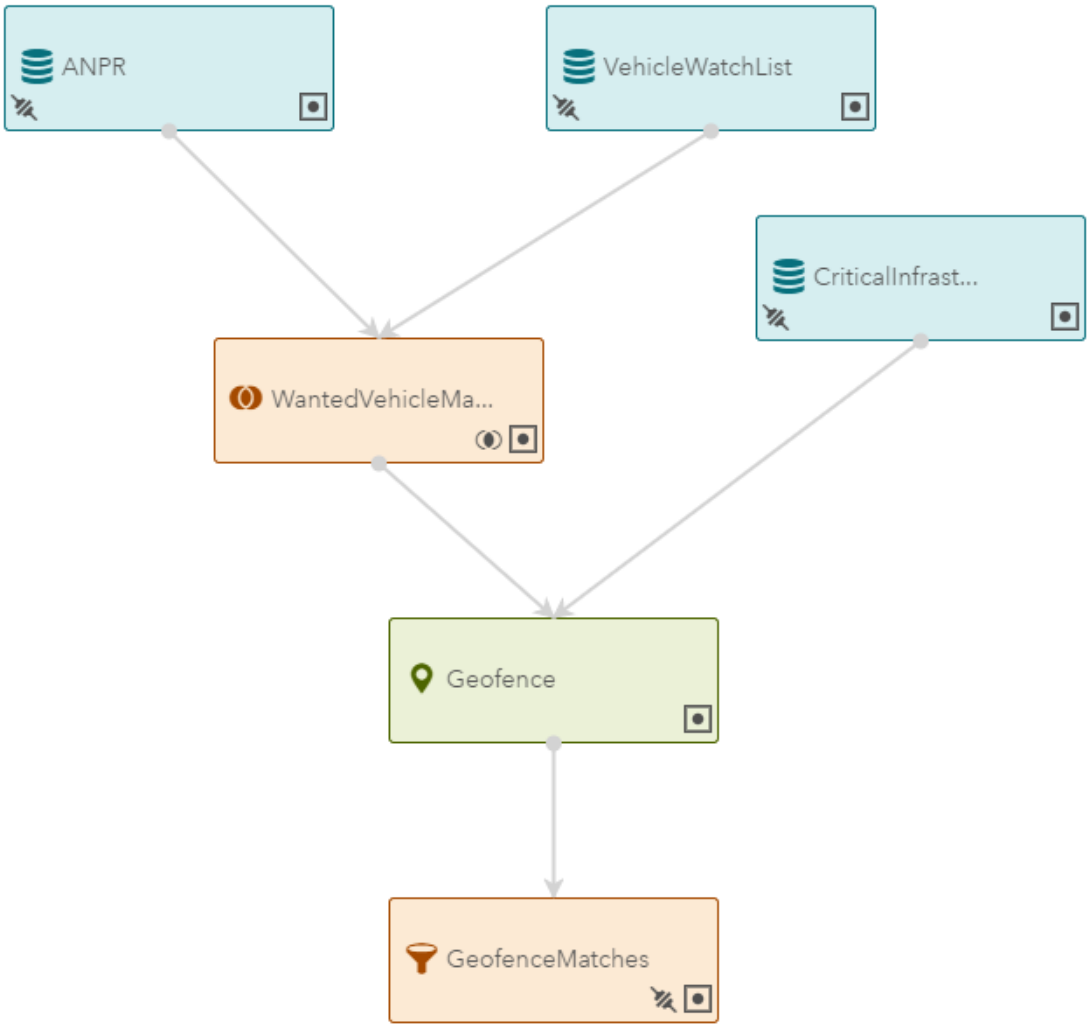
30 To stop the test, click ■.

Example: Geofence

This example creates a model that displays a list of wanted vehicles found in close proximity of critical infrastructure sites. The model performs the following actions:

- streams a list of vehicles, including vehicle locations
- streams a list of vehicles included on a vehicle watch list
- streams a list of critical infrastructure sites, including site locations
- processes the list of vehicles and attempts to match any wanted vehicles that are in close proximity to critical infrastructure sites
- produces a list of wanted vehicles found in close proximity to critical infrastructure sites

Diagram of the Geofence Model



Note: The CSV data and model XML code used in this example are available within your SAS Event Stream Processing installation, typically in the following location: `/opt/sas/viya/home/`

`SASEventStreamProcessingEngine/<release>/examples/xml/geofence2_xml`. Replace *<release>* with the release number in your SAS Event Stream Processing installation directory path.

1 On the **Projects** page, click New.

The New Project window appears.

2 In the New Project window, do the following:

a In the **Project name** field, enter **`geofence_demo`**.

b In the **Description** field, enter a description. Here is an example: **`This model can be used to identify wanted vehicles found in close proximity to critical infrastructure sites.`**

c Click **OK**.

If you do not currently have any test servers configured, you are prompted to decide whether you want to configure a test server now.

Note: It is assumed that you do not have any test servers configured. If you already have test servers configured, skip to step 5.

3 Click **Yes** to configure a test server now.

The Event Stream Processing Server window appears.

4 Configure a test server:

a In the **Name** field, enter a name to identify the new test server being created.

b In the **Host** field, enter the host name or IP address of the test server.

c In the **HTTP port** field, enter the test server's HTTP publish/subscribe port.

d If required, in the **Tags** field, enter any identifying keywords that describe the test server that you are creating.

e If required, change the setting for the **Authentication** field:

- **None**: This is the default option.
- **Authenticate using an OAuth token**: This option is relevant only if the ESP server is configured to require authorization. If you select this option, an additional field appears where you must enter the OAuth token.
- **Authenticate using a user name and password**: This option is relevant only if the ESP server is configured to require authorization. If you select this option, additional fields appear where you must enter the user name and password.

f If required, select the **Connect using SSL** check box. Selecting this option is relevant only if the test server is configured to require SSL encryption.

g Click **OK**.

Your project is created with a set of default properties.

5 Expand **Input Streams** on the **Windows** pane on the left and drag a Source window to the workspace.

The right pane displays the Source window's properties.

6 Specify a name and description for the Source window.

a In the right pane, in the **Name** field, change the default name to **`ANPR`**.

b In the **Description** field, enter `ANPR Source window`.

7 Configure the ANPR window to accept only “Insert” events and to automatically generate the key field:

a Expand **State and Event Type**.

b Select the **Accept only “Insert” events** check box.

c Select the **Automatically generate the key field** check box.

8 Specify an output schema for the ANPR window:

a In the right pane, click .

b Click .

The Output Schema window appears.

c Click to add a row to the schema table. After you add a row, click again to add the next row.

Enter the following values:

Key	Field Name	Type
N	vrm	String
N	lat	Double
N	long	Double
N	date	Timestamp
Y	pkey	String

d Click **OK**.

9 The ANPR window will stream a list of vehicles from a file called anpr.csv that contains example data. You can find this CSV file in the `geofence2_xml` folder in the `examples` directory. To add a connector to this CSV file:

a In the right pane, click .

b Expand **Input Data (Publisher) Connectors** and click .

The Publisher Connectors window appears.

c In the **Name** field, replace the default value with `anpr_csv_read`.

d In the **Fsname** field, enter the path to the CSV file. For example, you might enter `/opt/sas/viya/home/SASEventStreamProcessingEngine/<release>/examples/xml/geofence2_xml/anpr.csv`. Replace *<release>* with the release number in your SAS Event Stream Processing installation directory path.

e In the **Fstype** drop-down list, select **csv**.

f Configure the anpr_csv_read connector’s properties.

i Click **All properties**.

The All Properties window appears.

ii Enter `%Y-%m-%d %H:%M:%S` in the **Value** field of the **dateformat** property.

iii Enter `1` in the **Value** field of the **header** property.

iv Select **true** from the drop-down list in the **Value** field of the **ignorecsvparseerrors** property.

v Select **true** from the drop-down list in the **Value** field of the **noautogenfield** property.

vi Click **OK**.

g Click **OK**.

10 Expand **Input Streams** on the **Windows** pane on the left and drag another Source window to the workspace.

The right pane displays the Source window's properties.

11 Specify a name and description for the Source window.

a In the right pane, in the **Name** field, change the default name to `VehicleWatchList`.

b In the **Description** field, enter `VehicleWatchList Source window`.

12 Specify an output schema for the VehicleWatchList window:

a In the right pane, click .

b Click .

The Output Schema window appears.

c Click to add a row to the schema table. Enter the following values:

Key	Field Name	Type
Y	vrm	String

d Click **OK**.

13 The VehicleWatchList window will stream a list of wanted vehicles from a file called wantedvehicle.csv that contains example data. You can find this CSV file in the `geofence2_xml` folder in the `examples` directory. To add a connector to this CSV file:

a In the right pane, click .

b Expand **Input Data (Publisher) Connectors** and click .

The Publisher Connectors window appears.

c In the **Name** field, replace the default value with `vehicle_watchlist`.

d In the **Fsname** field, enter the path to the CSV file. For example, you might enter `/opt/sas/viya/home/SASEventStreamProcessingEngine/<release>/examples/xml/geofence2_xml/wantedvehicle.csv`. Replace *<release>* with the release number in your SAS Event Stream Processing installation directory path.

e In the **Fstype** drop-down list, select **csv**.

f Configure the vehicle_watchlist connector's properties.

i Click **All properties**.

The All Properties window appears.

ii Enter `1` in the **Value** field of the **header** property.

iii Click **OK**.

g Click **OK**.

14 Expand **Transformations** on the **Windows** pane on the left and drag a Join window to the workspace.

The right pane displays the Join window's properties.

15 Specify a name and description for the Join window.

a In the right pane, in the **Name** field, change the default name to `WantedVehicleMatch`.

b In the **Description** field, enter `WantedVehicleMatch Join window`.

16 Connect the ANPR window to the WantedVehicleMatch window with an edge.

a Position the cursor over the anchor point at the bottom of the ANPR window so that the anchor point changes from black to white.

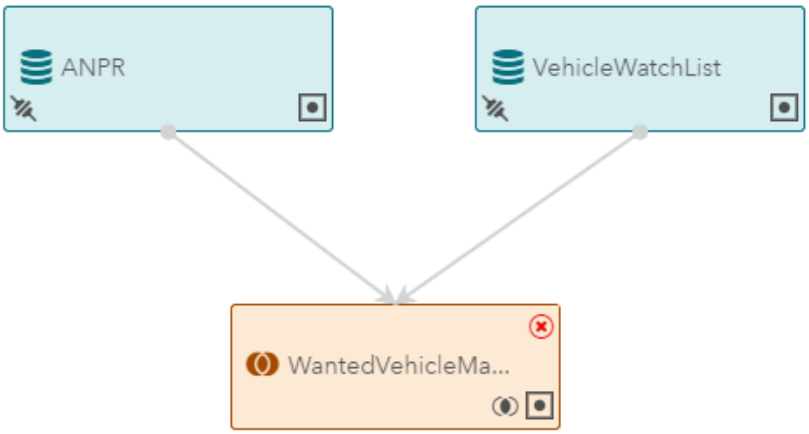
b Click the white anchor point, hold the mouse button down, and draw a line to the anchor point in the WantedVehicleMatch window.

The WantedVehicleMatch window now accepts values from the ANPR window.

17 Connect the VehicleWatchlist window to the WantedVehicleMatch window with an edge.

The WantedVehicleMatch window now accepts values from the VehicleWatchlist window.

Note: Each window in your model displays specific icons that represent window properties. For example, if a Source window contains a publisher connector, the window displays the corresponding publisher connector icon. For more information about window icons, see Window Icons on page 17. The WantedVehicleMatch window displays an error icon indicating that an invalid join type has been set. The occurrence of this error is expected behavior and will be resolved later when you set a valid join type.



18 Click the WantedVehicleMatch window in the workspace.

The right pane displays the Join window's properties.

19 Configure the calculation method for the WantedVehicleMatch window's output fields:

a In the right pane, expand **Settings**.

b Inspect the **Left window** and **Right window** fields. Notice that the ANPR window is regarded as the left window and the VehicleWatchList window is regarded as the right window. This is due to the order in which you added the edges.

c In the **Output field calculation method** field, select **Select fields** from the drop-down list.

d Collapse **Settings**.

20 Configure the WantedVehicleMatch window's join criteria:

a Expand **Join Criteria**.

b In the **Join Type** drop-down list, select **Inner**.

c Collapse **Join Criteria**.

21 Configure the WantedVehicleMatch window's join conditions:

a Expand **Join Conditions**.

b In the **Join Conditions** field, click to add a join condition.

c Click the cell in the Left: ANPR column, and select **vrm** from the drop-down list.

d Click the cell in the Right: VehicleWatchList column, and select **vrm** from the drop-down list.

22 Specify an output schema for the WantedVehicleMatch window:

a In the right pane, click .

b Click to add a row to the schema table.

The Output Schema window appears. Use this window to configure the fields as shown in the following table. The schema fields required have already been defined previously. Click to open the Copy Fields from Input Schema window. Select the following schema fields and click **OK**.

Field Name	Window	Fields	Type
vrm	ANPR	vrm	String
lat	ANPR	lat	Double
long	ANPR	long	Double
date	ANPR	date	TimeStamp

The Output Schema window displays the fields that you selected.

c Click **OK**.

23 Click .

24 Expand **Input Streams** on the **Windows** pane on the left and drag another Source window to the workspace.

The right pane displays the Source window's properties.

25 Specify a name and description for the Source window.

a In the right pane, in the **Name** field, change the default name to `CriticalInfrastructure`.

b In the **Description** field, enter `CriticalInfrastructure Source window`.

26 Specify an output schema for the CriticalInfrastructure window:

a In the right pane, click .

b Click .

The Output Schema window appears.

c Click to add a row to the schema table. After you add a row, click again to add the next row. Enter the following values:

Key	Field Name	Type
Y	name	String
N	lat	Double
N	long	Double
N	location	String
N	county	String
N	region	String
N	type	String
N	capacity	String
N	opened	String
N	closed	String
N	demolished	String
N	notes	String

d Click **OK**.

27 The CriticalInfrastucture window will stream a list of sites containing critical infrastructure from a file called infrastructure.csv that contains example data. You can find this CSV file in the `geofence2_xml` folder in the `examples` directory. To add a connector to this CSV file:

a In the right pane, click .

b Expand **Input Data (Publisher) Connectors**.

c Click .

The Publisher Connectors window appears.

d In the **Name** field, replace the default value with `infrastructure_csv_reader`.

e In the **Fsname** field, enter the path to the CSV file. For example, you might enter `/opt/sas/viya/home/SASEventStreamProcessingEngine/<release>/examples/xml/geofence2_xml/infrastructure.csv`. Replace <*release*> with the release number in your SAS Event Stream Processing installation directory path.

f In the **Fstype** drop-down list, select **csv**.

g Configure the infrastructure_csv_reader connector's properties:

i Click **All properties**.

The All Properties window appears.

ii Enter `1` in the **Value** field of the **header** property.

iii Select **true** from the drop-down list in the **Value** field of the **ignorecsvparseerrors** property.

iv Click **OK**.

h Click **OK**.

28 Expand **Utilities** on the **Windows** pane on the left and drag a Geofence window to the workspace.

The right pane displays the Geofence window's properties.

29 Specify a name and description for the Geofence window.

a In the right pane, in the **Name** field, change the default name to `Geofence`.

b In the **Description** field, enter `Geofence window`.

30 Connect the WantedVehicleMatch window to the Geofence window with an edge.

31 Connect the CriticalInfrastructure window to the Geofence window with an edge.

32 Click the Geofence window in the workspace.

The right pane displays the Geofence window's properties.

33 Configure the Geofence window's positional settings:

a Expand **Positions**.

b In the **Longitude** field, select **long** from the drop-down list.

c In the **Latitude** field, select **lat** from the drop-down list.

d In the **Default lookup distance (meters)** field, enter `100`.

e Collapse **Positions**.

34 Configure the Geofence window's geometric settings:

a Expand **Geometries**.

b In the table, in the **Longitude:** row, select **long** from the **Field** drop-down list.

c In the table, in the **Latitude:** row, select **lat** from the **Field** drop-down list.

d In the **Default radius (meters)** field, enter `100`.

e Collapse **Geometries**.

35 Configure the Geofence window's geofence algorithm properties:

a Expand **Geofence Algorithm Properties**.

b Select the **Record invalid geometries in the standard output log** check box.

c Collapse **Geofence Algorithm Properties**.

36 Configure the Geofence window's output map properties:

a Expand **Output Map**.

b In the **Geometry ID** field, enter `geoid`.

c In the **Event number** field, enter `eventnum`.

d Collapse **Output Map**.

37 Expand **Transformations** on the **Windows** pane on the left and drag a Filter window to the workspace.

The right pane displays the Filter window's properties.

38 Specify a name and description for the Filter window:

a In the **Name** field, change the default name to `GeofenceMatches`.

b In the **Description** field, enter `GeofenceMatches Filter window`.

39 Configure a subscribe connector for the GeofenceMatches window:

a Expand **Subscriber Connectors**.

b Click .

The Subscriber Connectors window appears.

c In the **Name** field, enter `sub`.

d Select the **Snapshot** check box.

e In the **Fsname** field, enter the path to the output file: **result.out**. For example, you might enter `/opt/sas/viya/home/SASEventStreamProcessingEngine/<release>/examples/xml/geofence2_xml/result.out`. Replace <*release*> with the release number in your SAS Event Stream Processing installation directory path.

f In the **Fstype** drop-down list, select **csv**.

g Click **OK**.

40 Specify a filter expression for the GeofenceMatches window:

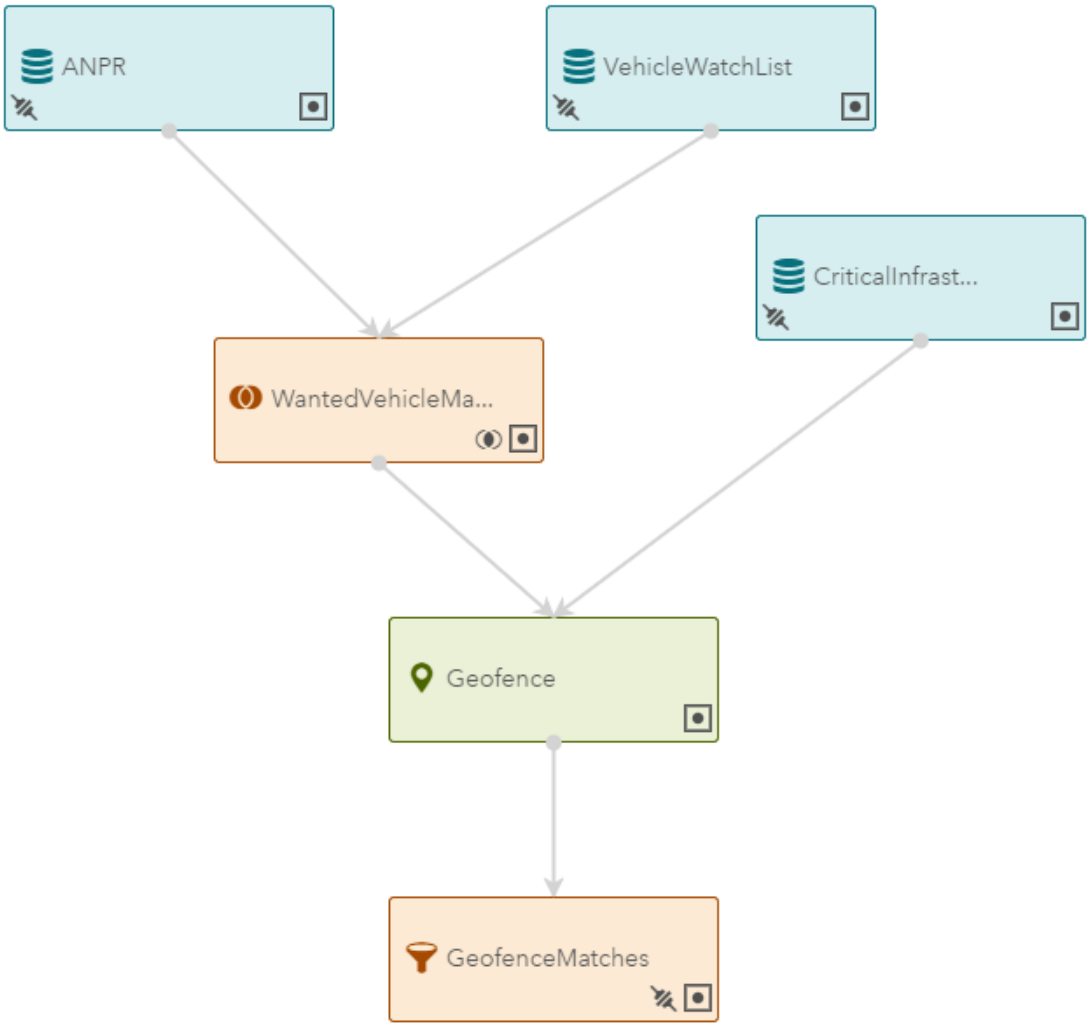
a Expand **Filter**.

b In the **Expression** field, enter `geoid != ''`

c Collapse **Filter**.

41 Connect the Geofence window to the GeofenceMatches window with an edge.

The Geofence Window Connected to the GeofenceMatches Window



42 Configure your model's connector orchestration:

a Click .

b In the right pane, expand **Connector Orchestration**.

c Click below the **Connector groups** label.

The Connector groups window appears.

d In the **Name** field, enter `sub1`.

e Click below the **Connectors** label.

f In the Connector column, click the newly created row and select **cq1/GeofenceMatches/sub** from the drop-down list.

g In the Target state column, select **Running** from the drop-down list.

h Click **OK**.

i Click below the **Connector groups** label.

The Connector groups window appears.

j In the **Name** field, enter `pub1`.

k Click below the **Connectors** label.

l In the Connector column, click the newly created row and select **cq1/ANPR/anpr_csv_read** from the drop-down list.

m In the Target state column, confirm that **Finished** is selected from the drop-down list.

n Click **OK**.

o Click below the **Connector groups** label.

The Connector groups window appears.

p In the **Name** field, enter `pub2`.

q Click below the **Connectors** label.

r In the Connector column, click the newly created row and select **cq1/CriticalInfrastructure/infrastructure_csv_reader** from the drop-down list.

s In the Target state column, confirm that **Finished** is selected from the drop-down list.

t Click **OK**.

u Click below the **Connector groups** label.

The Connector groups window appears.

v In the **Name** field, enter `pub3`.

w Click below the **Connectors** label.

x In the Connector column, click the newly created row and select **cq1/VehicleWatchlist/vehicle_watchlist** from the drop-down list.

y In the Target state column, confirm that **Finished** is selected from the drop-down list.

z Click **OK**.

aa Configure the dependency rules. Click below the **Dependency rules** label. After you add a row, click again to add the next row.

Enter the following values in the rows:

Row	Controlling Group	Dependent Group
1	sub1	pub1
2	pub2	pub1
3	pub2	pub3

ab In the right pane, click .

The XML Editor appears.

ac Locate the following line in the XML code: `<edge source="sub1" target="pub1"/>`

ad Amend this line to the following: `<edge source="sub1" target="pub1 pub2 pub3"/>`

43 Configure your model's threading level:

a In the right pane, click .

b In the right pane, expand **Attributes**.

c In the Threads field, enter `8`.

44 The model is now complete. Click to save your model.

45 Click ▸ Test .

A new page called **Test: geofence_demo** appears.

46 In the **Test Server** drop-down list, select the test server on which you want to test the model.

47 Click ▶.

The results for each window appear on separate tabs:

- the **cq1.ANPR** tab lists all vehicles within close proximity of critical infrastructure sites
- the **cq1.VehicleWatchlist** tab lists all vehicles on the vehicle watch list
- the **cq1.WantedVehicleMatch** tab lists the wanted vehicles found within close proximity of critical infrastructure sites
- the **cq1.Geofence** tab lists the geofencing information relating to the matched vehicles
- the **cq1.CriticalInfrastructure** tab lists sites containing critical infrastructure
- the **cq1.GeofenceMatches** tab shows wanted vehicles found within close proximity of critical infrastructure sites

Note: If the table is empty, check that the publisher connectors for the ANPR, VehicleWatchList, and CriticalInfrastructure windows are set correctly to point to the CSV files.

48 To stop the test, click ■.

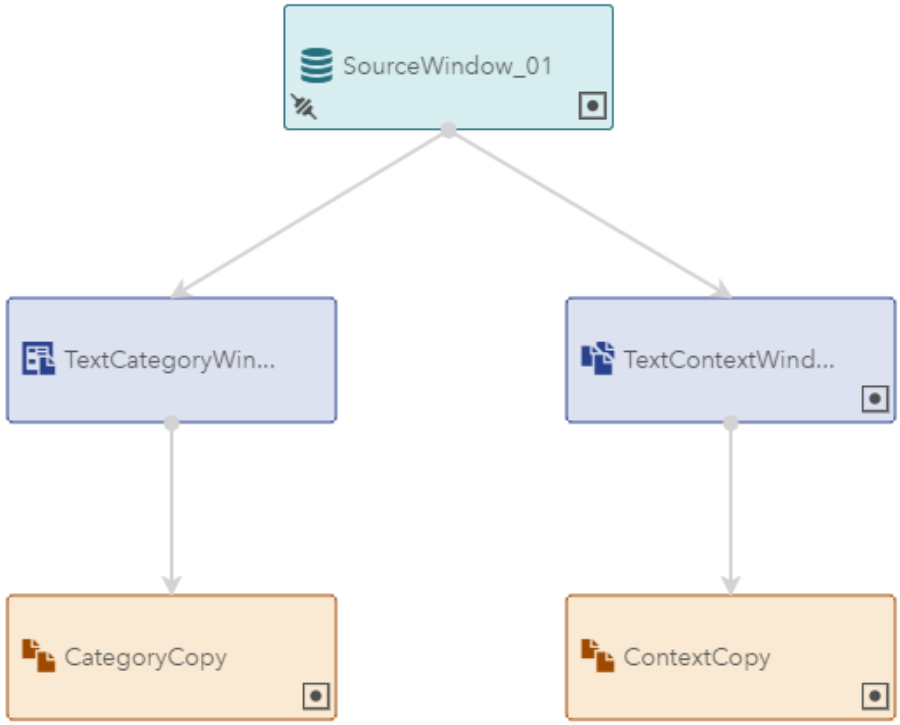
The project stops and then unloads from the ESP server.

Example: Working with Text Analytics

This example demonstrates how text can be analyzed, categorized, and sorted in context using SAS Event Stream Processing Studio. Text from a CSV file is processed and then cross-referenced against a set of text rules. Using the rules defined in an MCO file, the text is analyzed and categorized. The model also generates contextual information based on a set of rules defined in a LITI file.

Note: To successfully complete this example, you must have access to a SAS Contextual Analysis license.

Diagram of the Text Analytics Model



Note: The data files and model XML code used in this example are available from the following location: `ftp://ftp.sas.com/techsup/download/esp/esp-5.1.TextAnalyticsExample.zip`.

1 Download the esp-5.1.TextAnalyticsExample.zip file from `ftp://ftp.sas.com/techsup/download/esp/esp-5.1.TextAnalyticsExample.zip`.

2 Navigate to the esp-5.1.TextAnalyticsExample.zip file that you downloaded and extract its contents.

   Note: It is recommended that you extract the files into a folder called `text_Analytics`. Note the location that you extracted the files to.

3 On the **Projects** page, click New.

   The New Project window appears.

4 In the New Project window, do the following:

   a In the **Project name** field, enter `text_analytics_demo`.

   b In the **Description** field, enter a description. Here is an example: `This model uses predefined rules to analyze, categorize, and sort information in context.`

   c Click **OK**.

   If you do not currently have any test servers configured, you are prompted to decide whether you want to configure a test server now.

   Note: It is assumed that you do not have any test servers configured. If you already have test servers configured, go to step 7.

5 Click **Yes** to configure a test server now.

The Event Stream Processing Server window appears.

6 Configure a test server:

a In the **Name** field, enter a name to identify the new test server that you want to create.

b In the **Host** field, enter the host name or IP address of the new test server.

c In the **HTTP port** field, enter the new test server's HTTP publish/subscribe port.

d If required, in the **Tags** field, enter any identifying keywords that describe the new test server.

e If required, change the setting for the **Authentication** field:

- **None**: This is the default option.
- **Authenticate using an OAuth token**: This option is relevant only if the ESP server is configured to require authorization. If you select this option, an additional field appears where you must enter the OAuth token.
- **Authenticate using a user name and password**: This option is relevant only if the ESP server is configured to require authorization. If you select this option, additional fields appear where you must enter the user name and password.

f If required, select the **Connect using SSL** check box. Selecting this option is relevant only if the test server is configured to require SSL encryption.

g Click **OK**.

Your project is created with a set of default properties.

7 Configure your model's threading level:

a In the right pane, confirm that the project's properties appear. If they do not appear, click .

b Expand **Attributes**.

c In the **Threads** field, enter `4`.

8 Expand **Input Streams** on the **Windows** pane on the left and drag a Source window to the workspace.

The right pane displays the Source window's properties.

9 Specify a name and description for the Source window:

a In the right pane, in the **Name** field, change the default name to `SourceWindow_01`.

b In the **Description** field, enter `This window processes the event stream enabling the model's derived windows to analyze the text data`.

10 Specify an output schema for the SourceWindow_01 window:

a In the right pane, click .

b Click .

The Output Schema window appears.

c Click to add a row to the schema table. After you add a row, click again to add the next row.

Enter the following values:

Key	Field Name	Type

Y	ID	Int32
N	tstamp	Date
N	msg	String

d Click **OK**.

11 The SourceWindow_01 window will stream a list of vehicles from a file called textanalytics.csv that contains example data. To add a connector to this CSV file:

a In the right pane, click .

b Expand **Input Data (Publisher) Connectors** and click .

The Publisher Connectors window appears.

c In the **Name** field, replace the default value with `DataIn`.

d In the **Fsname** field, enter the path to the CSV file. For example, you might enter `/opt/sas/viya/home/SASEventStreamProcessingEngine/<release>/examples/xml/text_Analytics/textanalytics.csv`. Replace *<release>* with the release number in your SAS Event Stream Processing installation directory path.

e In the **Fstype** drop-down list, select **csv**.

f Configure the DataIn connector's properties:

i Click **All properties**.

The All Properties window appears.

ii Enter `%Y-%m-%d %H:%M:%S` in the **Value** field of the **dateformat** property.

iii Enter `100000` in the **Value** field of the **repeatcount** property.

iv Click **OK**.

g Click **OK**.

12 Expand **Text Analytics** on the **Windows** pane on the left and drag a Text Category window to the workspace.

The right pane displays the Text Category window's properties.

13 Specify a name and description for the Text Category window:

a In the right pane, in the **Name** field, change the default name to `TextCategoryWindow`.

b In the **Description** field, enter `This window processes text from a CSV file. The text is cross-referenced against a set of text rules in an MCO file. Using these rules, the text is analyzed and categorized.`

14 Connect the SourceWindow_01 window to the TextCategoryWindow window with an edge:

a Position the cursor over the anchor point at the bottom of the SourceWindow_01 window so that the anchor point changes from black to white.

b Click the white anchor point, hold the mouse button down, and draw a line to the anchor point in the TextCategoryWindow window.

The TextCategoryWindow window now accepts values from the SourceWindow_01 window.

15 Click the TextCategoryWindow window on the workspace.

16 Configure the TextCategoryWindow window's state and index:

a In the right pane, expand **State**.

b In the **Window state and index** field, select **Stateless (pi_EMPTY)** from the drop-down list.

c Collapse **State**.

17 Configure the TextCategoryWindow window's text categorization properties:

a Expand **Text Category** if the section is not expanded by default.

b In the **Text field** field, confirm that **msg** is selected by default.

c In the **Categorization binary MCO file (full pathname)** field, enter the path to your IPTC.mco file. For example, you might enter `/opt/sas/viya/home/SASEventStreamProcessingEngine/<release>/examples/xml/text_Analytics/IPTC.mco`. Replace <*release*> with the release number in your SAS Event Stream Processing installation directory path.

18 Expand **Transformations** on the **Windows** pane on the left and drag a Copy window to the workspace.

The right pane displays the Copy window's properties.

19 Specify a name and description for the Copy window:

a In the right pane, in the **Name** field, change the default name to `CategoryCopy`.

b In the **Description** field, enter `This window retains categorized events for five minutes.`

20 Connect the TextCategoryWindow window to the CategoryCopy window with an edge.

21 Configure the CategoryCopy window's state and index:

a Click the CategoryCopy window on the workspace.

b Expand **State**.

c In the **Window state and index** field, select **Stateful (pi_RBTREE)** from the drop-down list.

22 Configure the CategoryCopy window's retention properties:

a Expand **Retention** if the section is not already expanded by default.

b Select the **Limit event retention** check box.

c Confirm that the **Type** field is set to **By time, sliding**.

d In the **Time limit** field, enter `5` and select **Minutes** from the drop-down list.

e Collapse **Retention**.

23 Expand **Text Analytics** on the **Windows** pane on the left and drag a Text Context window to the workspace.

The right pane displays the Text Context window's properties.

24 Specify a name and description for the Text Context window:

a In the right pane, in the **Name** field, change the default name to `TextContextWindow`.

b In the **Description** field, enter `This window outputs information in context based on a set of rules defined in a LITI file.`

25 Connect the SourceWindow_01 window to the TextContextWindow window with an edge.

26 Configure the TextContextWindow window's text categorization properties:

a Click the TextContextWindow on the workspace.

b Expand **Text Context** if the section is not expanded by default.

c In the **Text field** field, confirm that **msg** is selected by default.

d In the **LITI files (full pathnames)** field, click .

A new row is created in the table.

e Click the new row and enter the path of your LITI file. For example, you might enter `/opt/sas/viya/home/SASEventStreamProcessingEngine/<release>/examples/xml/text_Analytics/citng_en-ne.li`. Replace <*release*> with the release number in your SAS Event Stream Processing installation directory path.

f Select the **Generate a null event if no match is found** check box.

27 Expand **Transformations** on the **Windows** pane on the left and drag a Copy window to the workspace.

The right pane displays the Copy window's properties.

28 Specify a name and description for the Copy window:

a In the right pane, in the **Name** field, change the default name to `ContextCopy`.

b In the **Description** field, enter `This window retains events that have been sorted in context for five minutes.`

29 Connect the TextContextWindow window to the ContextCopy window with an edge.

30 Configure the ContextCopy window's retention properties:

a Click the ContextCopy window in the workspace.

b Expand **Retention**.

c Select the **Limit event retention** check box.

d Confirm that the **Type** field is set to **By time, sliding**.

e In the **Time limit** field, enter `5` and select **Minutes** from the drop-down list.

31 Configure the project's continuous query:

a Click .

b In the **Name** field, enter `contquery_01`.

c Expand **Debugging**.

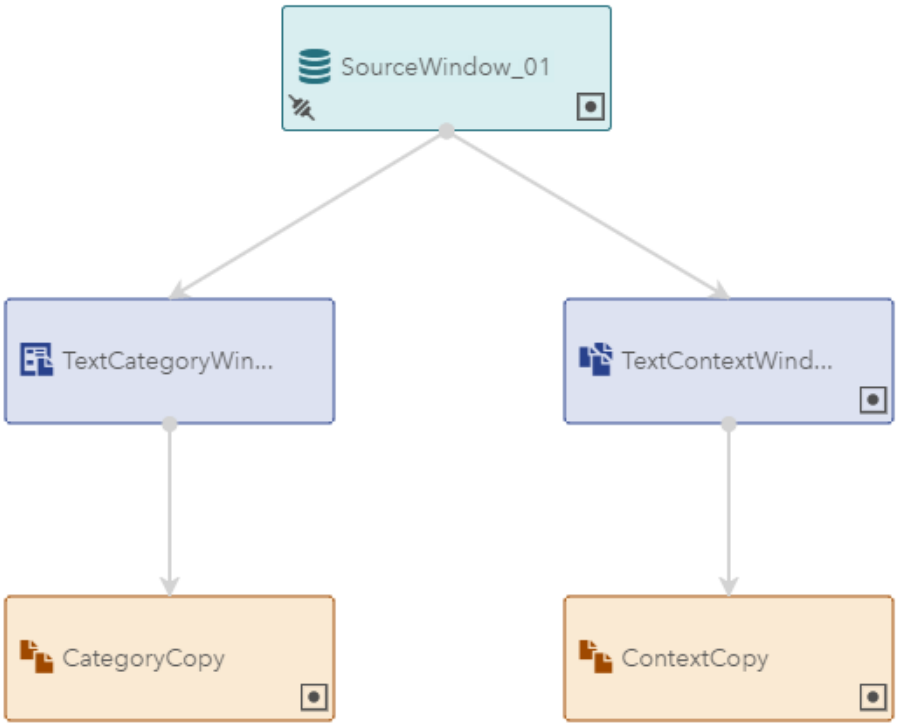
d Select the **Log warnings for long computation times** check box.

e In the **Threshold (µs)** field, enter `100`.

f In the **Trace in server log** field, select **TextCategoryWindow** and **TextContextWindow**.

32 Click .

Diagram of the Text Analytics Model



33 Click ▸ Test .

A new page called **Test: text_analytics_demo** appears.

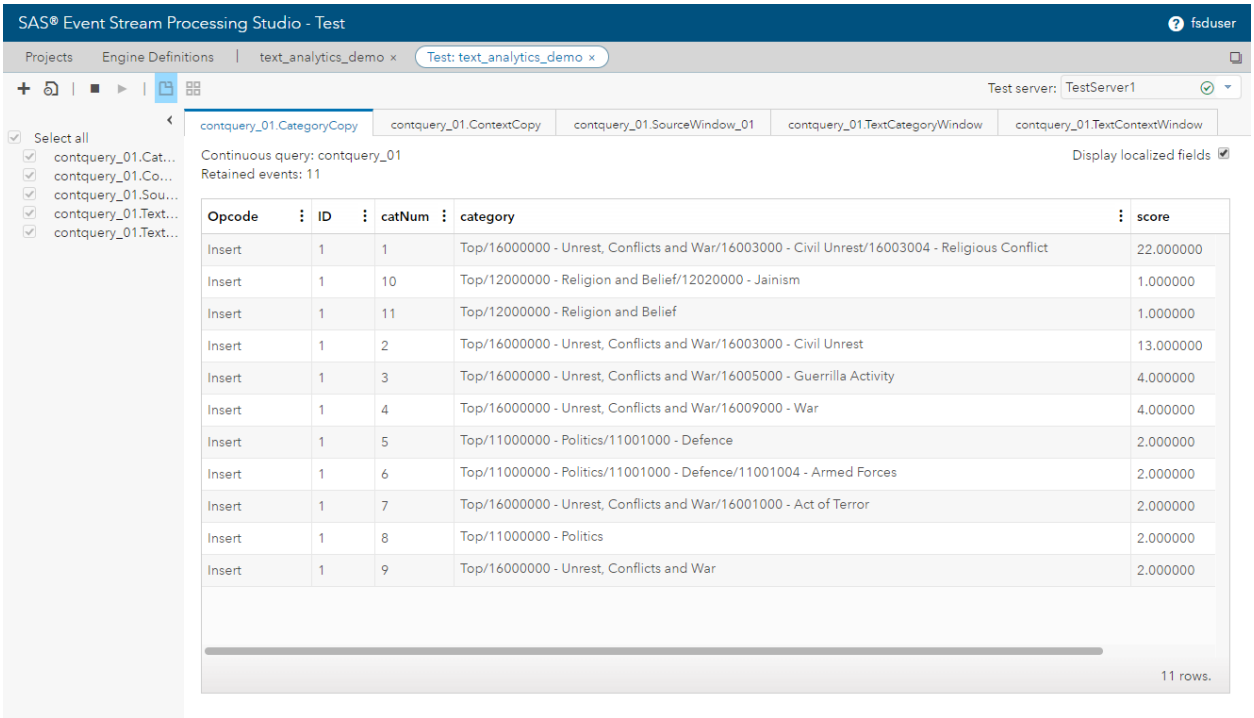
34 In the **Test Server** drop-down list, select the test server on which you want to test the model.

35 Click ▶.

The results for each window appear on separate tabs:

- the **contquery_01.CategoryCopy** tab displays categorized events with a retention policy of 5 minutes, as shown in the following figure:

Results for the CategoryCopy Window



Opcode	ID	catNum	category	score
Insert	1	1	Top/16000000 - Unrest, Conflicts and War/16003000 - Civil Unrest/16003004 - Religious Conflict	22.000000
Insert	1	10	Top/12000000 - Religion and Belief/12020000 - Jainism	1.000000
Insert	1	11	Top/12000000 - Religion and Belief	1.000000
Insert	1	2	Top/16000000 - Unrest, Conflicts and War/16003000 - Civil Unrest	13.000000
Insert	1	3	Top/16000000 - Unrest, Conflicts and War/16005000 - Guerrilla Activity	4.000000
Insert	1	4	Top/16000000 - Unrest, Conflicts and War/16009000 - War	4.000000
Insert	1	5	Top/11000000 - Politics/11001000 - Defence	2.000000
Insert	1	6	Top/11000000 - Politics/11001000 - Defence/11001004 - Armed Forces	2.000000
Insert	1	7	Top/16000000 - Unrest, Conflicts and War/16001000 - Act of Terror	2.000000
Insert	1	8	Top/11000000 - Politics	2.000000
Insert	1	9	Top/16000000 - Unrest, Conflicts and War	2.000000

- the **contquery_01.ContextCopy** tab displays contextual events with a retention policy of 5 minutes.
- the **contquery_01.TextContextWindow** tab displays information in contextual format.
- the **contquery_01.SourceWindow_01** tab displays the processed event stream.
- the **contquery_01.TextCategoryWindow** tab displays information in categorized format.

36 To stop the test, click ■.

The project stops and then unloads from the test server.

Working with SAS Micro Analytic Service Modules in SAS Event Stream Processing Studio

You can use SAS Micro Analytic Service modules to create input handler functions in SAS Event Stream Processing Studio using Python, DS2, and C.

SAS Event Stream Processing projects can also reference models that are stored in SAS Model Manager. When a project is deployed, the model is retrieved from SAS Model Manager and written to the ESP server. SAS Micro Analytic Service modules are used to accommodate the imported content that was created in SAS Model Manager. The module is uploaded and then referenced from the model's Procedural window's input handler. For more information, see "Example: Importing a Model Created in SAS Model Manager into SAS Event Stream Processing Studio" on page 61.

Create a SAS Micro Analytic Service Module

To create a new module:

1 Open your project and click .

2 In the right pane, expand **SAS Micro Analytic Service Modules**.

3 Click .

The SAS Micro Analytic Service Module window appears.

4 In the **Name** field, enter a name for the module.

5 In the **Language** drop-down list, select the language that you want to use to write the module.

6 In the **Description** field, enter a description of the module.

7 In the **Function names** field, enter a comma-separated list of function names.

8 In the **Code source** field, select one of the following options:

- **Embedded code** to enter your own code
- **External file** to use code located in an external file
- **SAS Micro Analytic Service store** to use code in an analytic store (ASTORE) file

9 If you selected **Embedded code** in the **Code source** field, enter your code in the **Embedded code** field.

10 If you selected **External file** in the **Code source** field, enter the file path to the external file in the **External file** field.

11 If you selected **SAS Micro Analytic store** in the **Code source** field:

a In the **External file** field, enter the file path to the ASTORE file.

b In the **SAS Micro Analytic Service store** field, enter the module store location.

c In the **SAS Micro Analytic Service store version** field, enter the version of the module store location.

d In the **Module Members** field, enter your module's member names. To add a new module member, click and fill in the applicable fields.

12 Click **OK**.

The module that you created appears in the SAS Micro Analytic Service Modules table.

Upload a SAS Micro Analytic Service Module

To upload an existing module:

1 Open your project and click .

2 In the right pane, expand **SAS Micro Analytic Service Modules**.

3 Click .

The Import a SAS Micro Analytic Service Module from SAS Model Manager window appears.

4 In the **ZIP file** field, click **Choose File**.

5 Select the SAS Model Manager ZIP file that you want to upload and click **Open**.

The Import a SAS Micro Analytic Service Module from SAS Model Manager window reloads to display information about the module's roles.

6 Review the module's role properties and modify them if necessary.

7 To copy the input schema for future use, click **View input schema** and then copy the input schema to your clipboard. To copy the output schema for future use, click **View output schema** and then copy the output schema to your clipboard.

8 Click **OK.**

The module that you uploaded appears in the SAS Micro Analytic Service Modules table.

Delete a SAS Micro Analytic Service Module

To delete a module, select the module that you want to delete from the SAS Micro Analytic Service Modules table and click .

The module is deleted from the SAS Micro Analytic Service Modules table.

Example: Importing a Model Created in SAS Model Manager into SAS Event Stream Processing Studio

This example demonstrates how to import a model from SAS Model Manager to process input data. SAS Event Stream Processing projects can reference models that are stored in SAS Model Manager. When a project is deployed, the model is retrieved from SAS Model Manager and written to the ESP server. SAS Micro Analytic Service modules are used to accommodate the imported content that was created in SAS Model Manager. The module is uploaded and then referenced from the model's Procedural window's input handler. Here is an example of a DS2 code file that might be imported from SAS Model Manager to SAS Event Stream Processing Studio:

```
ds2_options sas;
package module_1/overwrite=yes;
    method score(int quantity, double price, in_out int volume);
    volume = quantity * price;
    end;
endpackage;
```

This example code generates the volume of a set of stock market trades.

Note: A pre-configured model is not provided with this example. To successfully complete this example, you must provide your own champion model and data files.

Before you start, ensure that your deployment has SAS Model Manager installed and that the model that you plan to import is correctly configured in SAS Model Manager. You must also ensure that you have noted the order and type of your module's output fields.

To successfully import a model created in SAS Model Manager into a specific Procedural window in your model, the model that you want to import must meet the following criteria:

- The model is a champion of the project.
- The model has a score code type of dataStep, ds2Package, or dsMultiType that meets the following criteria:
  - The dataStep model contains a score file with a file role of score code.

- The ds2Package model contains a score file with a file role of score code.
- The dsMultiType file contains ASTORE content with a DS2 score code file named dmcas_packagescorecode.sas.

- The model is a single DS2 code file (DS2 Package) or a DS2 code file with one or more analytic store files.

1 On the **Projects** page, click New.

The New Project window appears.

2 In the New Project window, do the following:

a In the **Name** field, enter `MM_Import`.

b In the **Description** field, enter: `This example demonstrates how to import a champion model from SAS Model Manager`.

c Click **OK**.

It is assumed that you already have a test server configured.

3 Click .

4 In the right pane, configure your project's properties. Update the fields as required.

5 Expand **Input Streams** on the **Windows** pane on the left and drag a Source window to the workspace.

The right pane displays the Source window's properties.

6 Expand **Utilities** on the **Windows** pane on the left and drag a Procedural window to the workspace.

The right pane displays the Procedural window's properties.

7 Connect the Source window to the Procedural window with an edge.

a Position the cursor over an anchor point in the Source window so that the anchor point changes from black to white.

b Click the white anchor point, hold the mouse button down, and draw the line to an anchor point in the Procedural window.

The Procedural window now accepts events from the Source window.

8 Click the Procedural window in the workspace.

9 Expand **Input Handlers**.

a Click the row containing the Source window.

b Click .

The Input Handlers window appears.

c In the **Handler Type** field, select **Import from SAS Model Manager**.

The Select Model window appears, showing SAS Model Manager repositories in a collapsed state.

d Navigate to the SAS Model Manager project containing the model that you want to import.

e Select the model that you want to import.

The window refreshes to display additional information about the model that you have selected to import.

f Inspect the additional information about the model.

g If necessary, you can add the input schema fields and output schema fields that you are importing to the window. You can also copy the input schema fields and output schema fields for future use. To add the input schema, select **Add input schema to window** and select the window that you want to import the schema to from the list. To add the output schema, select **Add output schema to window** and select the window that you want to import the schema to from the list.

   Alternatively, to copy the input schema for future use, click **View input schema** and then copy the input schema to your clipboard. To copy the output schema for future use, click **View output schema** and then copy the output schema to your clipboard.

h Click **OK**.

   The Input Handlers window appears.

   Your model's XML code is updated to reference the module that you imported. The Input Handlers window displays the name of the imported module and the function to be called from the Procedural window.

i Inspect and, if necessary, modify the information in the Input Handlers window.

j Click **OK**.

   The import is completed. The imported code is written to the ESP server, typically to the `/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/mas-modules` directory. This directory might be different depending on how your deployment has been configured.

10 Configure a publisher connector:

   Note: It is recommended that you note the name, order, and type of the fields defined in your publisher connector's input file. The name, order, and type of the fields defined in your input file must match the name, order, and type of the fields defined in your Source window's output schema.

a Click the Source window in the workspace.

b Expand **Input Data (Publisher) Connectors** and click .

   The Publisher Connectors window appears.

c In the **Name** field, enter `Connector`.

d Configure the remaining fields as required.

e Click **OK**.

11 Configure the output schema:

a Click the Source window in the workspace.

b In the right pane, click .

c Click .

   The Output Schema window appears.

d Inspect your output schema to establish if a key field is defined. If a key field is not defined, define a key field.

e Inspect your output schema to establish if its fields match the name, order, and type of the output fields defined in the SAS Micro Analytic Service module. Your output schema fields must also match the name, order, and type of the fields defined in your input file. These fields must match for the imported model to run successfully in test mode.

12 Click .

13 Click ▶ Test.

A new page called **Test: MM_example** appears.

14 Click ▶.

The results for each window appear in separate tabs.

15 To stop the test, click ■.

After the import has completed, your model's XML code is updated to display the following information relating to the created SAS Micro Analytic Service module:

- A `<metadata>` element containing a set of unique identifiers. These identifiers are required to obtain the model's content when you run the model in test mode. Here is an example of an imported model's `<metadata>` element:

```
<metadata>
    <meta id="layout">{"trades_traders_cq":{"Trades":{"x":50,"y":50},"pw1":{"x":50,"y"
    :171.99652862548828}}}</meta>
    <meta id="mm_linked_module_1">fae64eb3-2de3-45ac-931a-088a56a49062,83455da4-5b04-41bd
    -944e-a5c1b2bf63cb,ds2Package</meta>
</metadata>
```

  Note: If you change the content of the `<metadata>` element, SAS Event Stream Processing Studio might be unable to locate your model's content.

- A `<mas-modules>` element containing identifying information about the module that you imported. Here is an example of an imported model's `<mas-modules>` element:

```
<mas-modules>
       <mas-module module="module_1" language="ds2" func names="score" mas-store="fae64eb3-2de3-45ac
    -931a-088a56a49062_champion_45c08c8c-8490-4646-e8e518430379" mas-store-version="1.0">
    <code-file><![CDATA[score.as.ds2]]</code-file>
</mas-modules
```

  In the example code here, the `<mas-module>` element defines an input handler to the SAS Micro Analytic Service engine. The `<mas-store>` attribute contains the SAS Micro Analytic Server store name. The store name is `fae64eb3-2de3-45ac-931a-088a56a49062`, the ID for the ESP project is `45c08c8c-8490-4646-e8e518430379`, and the store version number is `1.0`. The model type is also specified as `champion`.

  The `<code-file>` element encloses the name of the score.as.ds2 file that contains the code used as an input handler.

  Because the imported code is written to the ESP server, not the model, you can only view the code from the ESP server. In the example code here, the imported code is written to the /opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/mas-modules/fae64eb3-2de3-45ac-931a-088a56a49062_champion_45c08c8c-8490-4646-e8e518430379/1.0/score.mas.ds2 file. This file's location might be different depending on how your deployment has been configured.

- New schema fields for the Source and Procedural windows (if the model that you imported contained additional schema fields).

- A `<mas-map>` element in the Procedural window referencing the newly created SAS Micro Analytic Service module and its function. The `<mas-map>` element binds the function defined in the `<mas-module>` element to the input stream in your model's Procedural window. Here is an example of an imported model's `<mas-map>` element:

```
<mas-map>
    <window-map module="module_1" function="score" revision="0" source="Trades"/>
</mas-map>
```

Working with Input Handlers

You can use score code to create input handler functions in SAS Event Stream Processing Studio using Python, DS2, and C. You can also import existing score code into SAS Event Stream Processing Studio from SAS Model Manager.

Creating Input Handler Functions in Procedural Windows

To create an input handler function within a Procedural window:

1. Open the relevant project.
2. Click the relevant Procedural window.

   The right pane displays the properties of the Procedural window.
3. In the right pane, expand **Input Handlers**.
4. Click the row for the input window that you want to link the import handler function to.
5. Click .

   The Input Handler window appears.
6. In the **Handler type** field, select one of the following handler types:
   - DS2 code – enables you to enter DS2 code directly
   - DS2 code file – enables you to reference an external file containing DS2 code
   - Plug-in – enables you to reference a plug-in library
   - DS external – enables you to enter DATA step code directly

     Note: When you configure a model that contains a Procedural window that executes DATA step code, you must add the `ds-initialize` element to your project using the XML Editor.
   - DS external file – enables you to reference an external file containing DATA step code

     Note: When you configure a model that contains a Procedural window that executes DATA step code, you must add the `ds-initialize` element to your project using the XML Editor.
   - SAS Micro Analytic Service – enables you to reference a SAS Micro Analytic Service module

   The Input Handlers window reloads to display fields relating to the handler type that you selected.
7. Update any other fields as required.
8. Click **OK**.

   The Input handler functions section refreshes to display the imported function.

You can use SAS Event Stream Processing Studio to specify a function that returns derived context using the cxx-plugin-context plug-in. To do this:

1. Open your model and click the relevant Procedural window on the workspace.

   The right pane displays the properties of the Procedural window.

2 In the right pane, expand **Input Handlers**.

3 Select the **Use a context plug-in** check box.

4 In the **Plug-in library name** field, enter a name for the relevant plug-in library.

5 In the **Function** field, enter the function name.

6 To create a property list, in the **Property map** field, click to add a new row to the table. Click again to add an additional row to the table:

   a In the **Name** field, enter a name for the property.

   b In the **Value** field, enter a value for the property.

Importing Score Code Created in SAS Model Manager to Procedural Windows

You can import score code created in SAS Model Manager directly into a specific Procedural window in your model. The imported score code is directly written to the Procedural window's input handler.

To import score code directly into a Procedural window:

1 Open the project that you want to import the score code to.

2 Click the relevant Procedural window.

   The right pane displays the properties of the Procedural window.

3 In the right pane, expand **Input Handlers**.

4 Click the row for the input window that you want to link the import handler function to.

5 Click .

   The Input Handler Functions window is displayed.

6 In the **Handler type** field, select **Import score code from SAS Model Manager**.

7 Click **Choose File**.

8 Navigate to the file containing the SAS Model Manager ZIP file that you want to upload and click **Open**.

9 Inspect and, if necessary, modify the information in the Roles table.

10 It is recommended that you add the input schema that you are importing to a window or copy the input schema for future use. To add the input schema to a window, select **Add input schema to window** and select the window that you want to import the schema to from the list. Alternatively, to copy the input schema for future use, click **View input schema** and then copy the input schema to your clipboard.

11 Click **OK**.

   The Input Handlers window appears.

12 Inspect and, if necessary, modify the information in the Input Handler Functions window.

13 Click **OK**.

   The Input handler functions section refreshes to display the imported function.