SAS® Event Stream Processing 4.3: 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 XML server. Run the following command from your terminal:

```
$DFESP_HOME/bin/dfesp_xml_server -pubsub n -http-admin adminport
```

2

```
-http-pubsub pubsubport
```

Where *n* specifies the publish/subscribe port, *adminport* specifies the admin port, and *pubsubport* specifies the HTTP publish/subscribe port. The terminal log should confirm successful server instantiation and reiterate the three ports specified here. For information about the XML server, see "Using the ESP Server" in SAS Event Stream Processing: Using the XML Layer.

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.

Accessing SAS Event Stream Processing Studio

To access SAS Event Stream Processing Studio:

Open the following URL:

`https://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, this port is 8080.

If you successfully access SAS Event Stream Processing Studio, the SAS Event Stream Processing Studio home page is displayed.

Understanding the User Interface

Pages

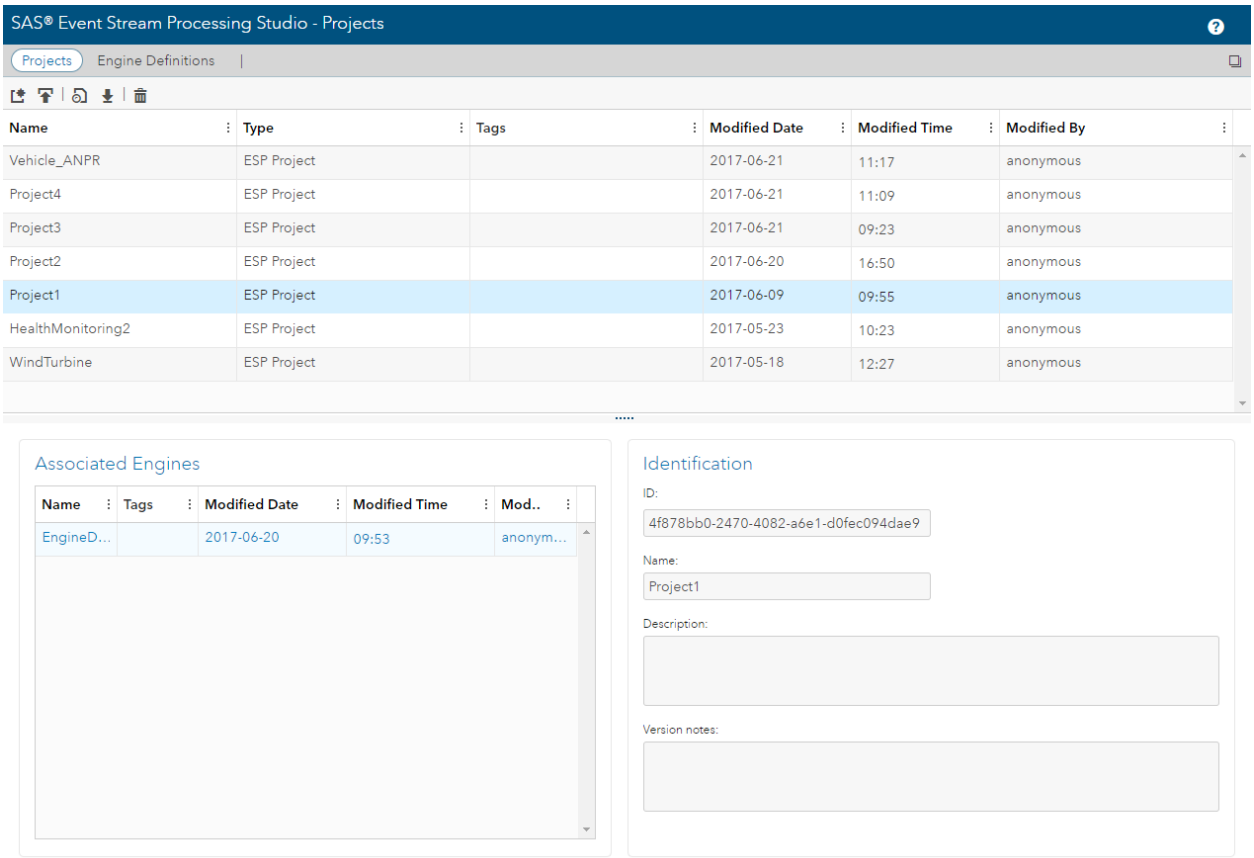
A *page* is the highest level container in the user interface. All other user interface elements are contained within the confines of a page.

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 is displayed.

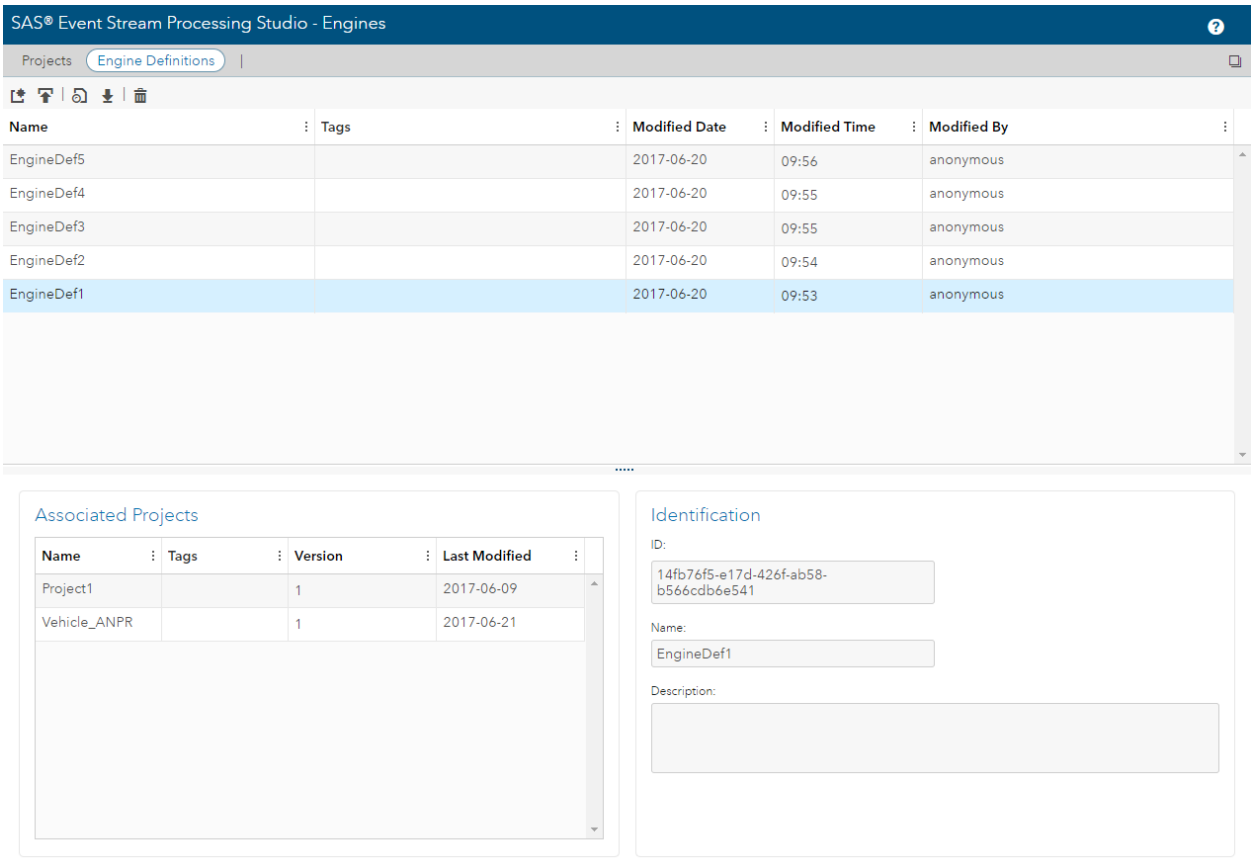
The Projects Page displaying active test projects



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 two panes: **Associated Projects** and **Identification**.

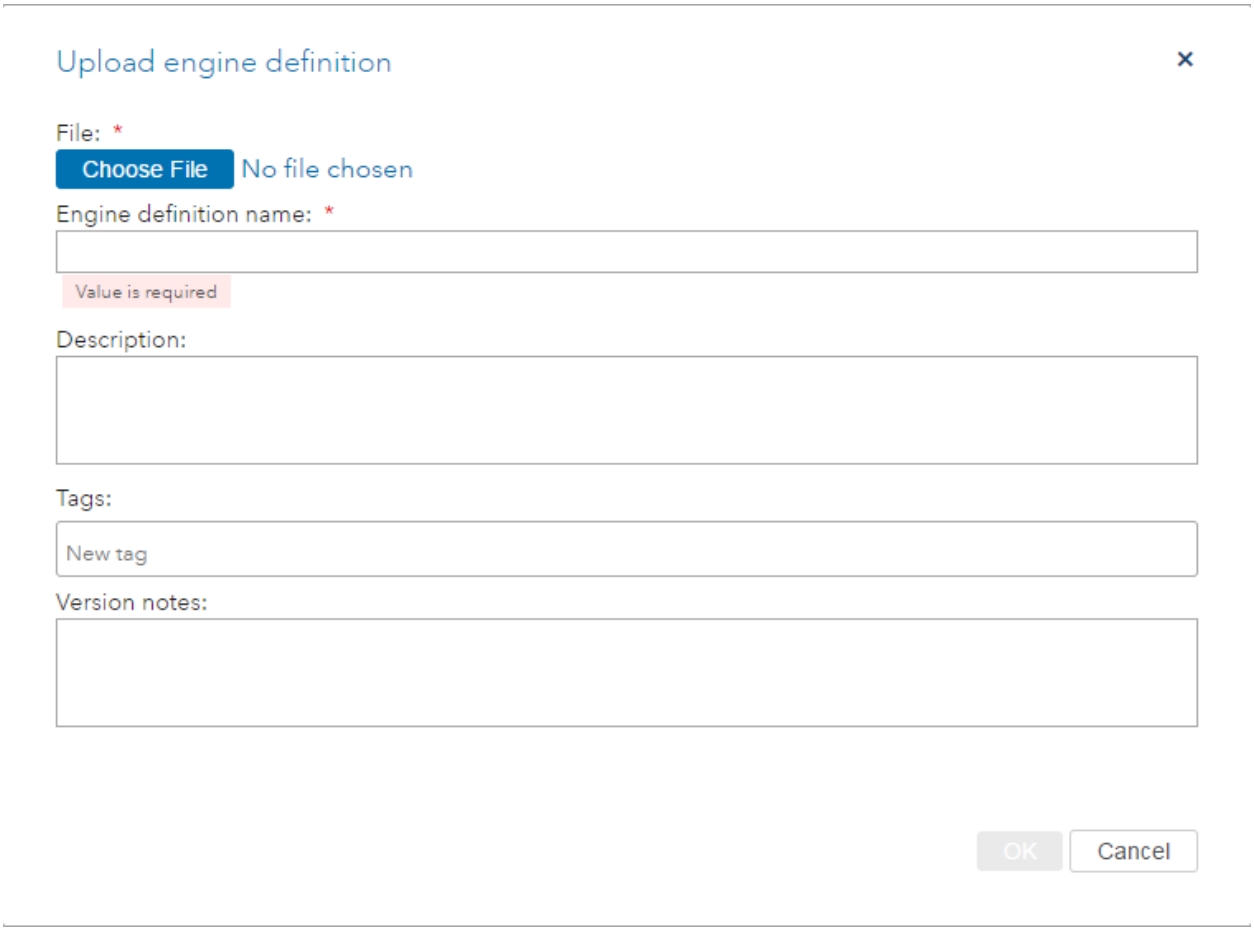
The Engine Definitions Page Contains Two Panes: Associated Projects and Identification



Windows

A *window* is a floating user interface element that is often displayed 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.

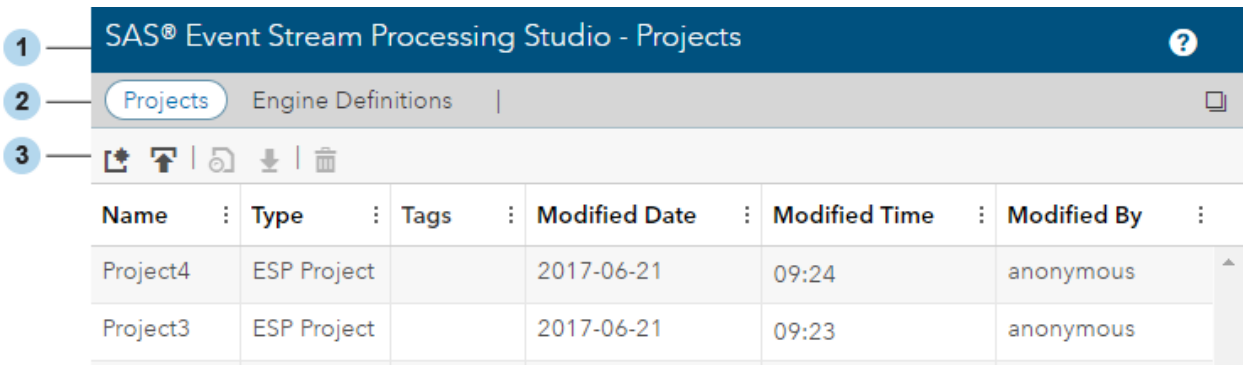
The Upload Engine Definition Window



Toolbars

There are three main toolbars in SAS Event Stream Processing Studio, as shown in the following figure. For information about each toolbar, see the following table:

SAS Event Stream Processing Studio Toolbars



Item Number	Description	Name
1	Application bar	■ states whether you are currently viewing projects or engine definitions ■ provides access to Help and product information
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

Sometimes a large number of items, and, as a result, a large amount of information, is displayed on pages and panes. To make it easier to work with a large amount of information, you can sort and filter items displayed in grids. You can also show, hide, and re-order 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 twice. 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 is displayed. 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.

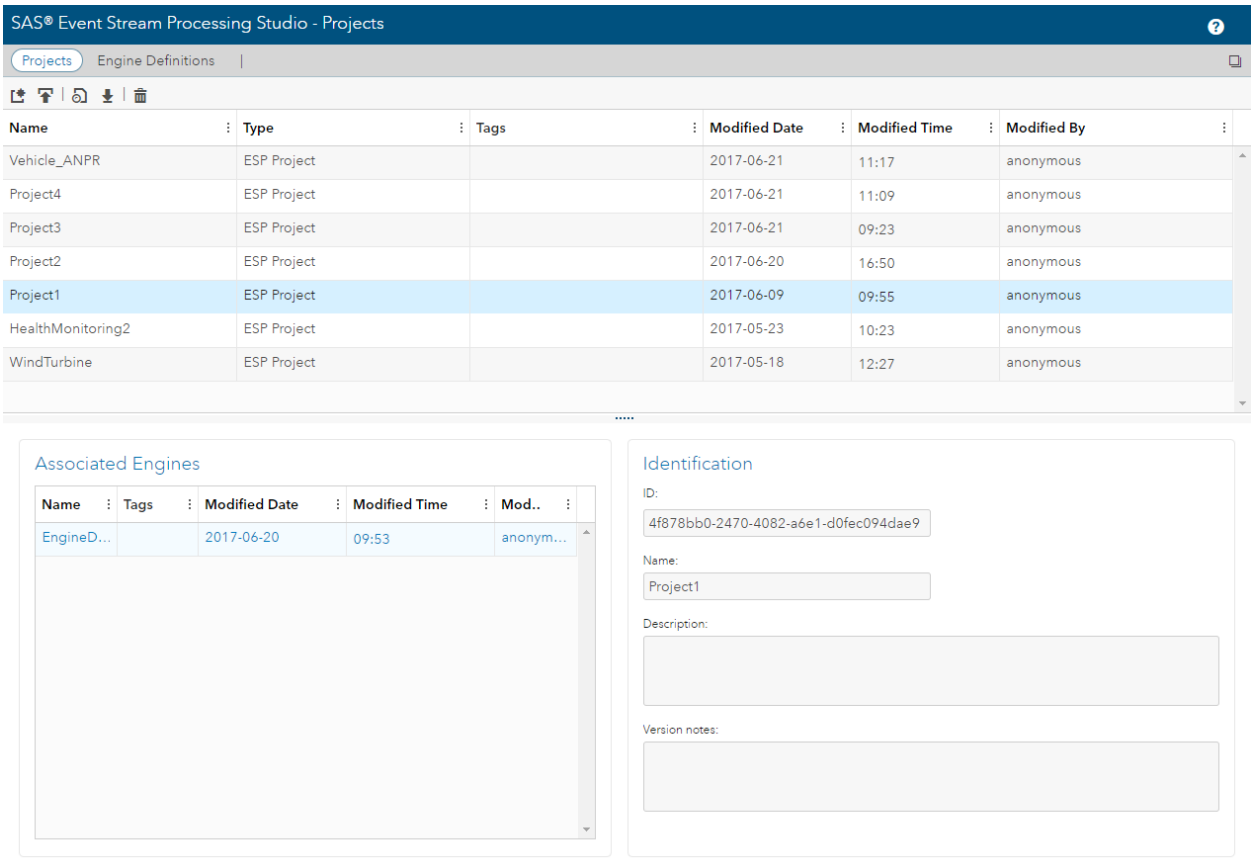
For more information about the modeler, see Using SAS Event Stream Processing Studio Modeler on page 11. For more information about the XML Editor, see Using the XML Editor on page 17.

Working with Projects in SAS Event Stream Processing Studio

A *project* is composed 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.

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.



Creating Projects

To create a new project:

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

SAS Event Stream Processing Modeler is displayed. However, if you do not currently have any ESP servers configured, the Event Stream Processing Server window is displayed first.

2 If the Event Stream Processing Server window is displayed, do the following:

a In the **Name** field, enter a name to identify the new ESP server being added.

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

c In the **Admin Port** field, enter the test server's administration port number.

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

e If required, enter a value in the **OAuth** field. This field is relevant only if the SAS Event Stream Processing engine is configured to require authorization for use.

f If required, select the **SSL** check box. Selecting this check box is relevant only if the SAS Event Stream Processing engine is configured to require SSL encryption.

g Click **OK**.

Note: To register additional test servers, click  . To display the details of the test server that you have selected, click  .

3 Click  .

SAS Event Stream Processing Modeler is displayed.

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**.

4 In the **Project name** field, enter a name for the project.

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

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

7 In the **Version notes** field, enter any notes relating to the initial version of the project.

Note: This version of SAS Event Stream Processing Studio does not enable you to increment the version number of a project. To create a new version of your project, you must create an empty project.

8 Click **OK**.

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 Open your project and click  .

b Review the default project properties and modify them if necessary.

c You can also add or modify additional project properties, such as SAS Micro Analytic Service modules, property lists, and connector orchestration.

Uploading Projects

To upload an existing project:

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

The Upload Project window is displayed.

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

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

4 In the **Project 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 In the **Version notes** field, enter any notes relating to the initial version of the project.

Note: This version of SAS Event Stream Processing Studio does not enable you to increment the version number of a project.

8 Click **OK**.

The uploaded project is displayed on the **Projects** page.

Downloading Projects

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

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 grid on the **Projects** page and click  .

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

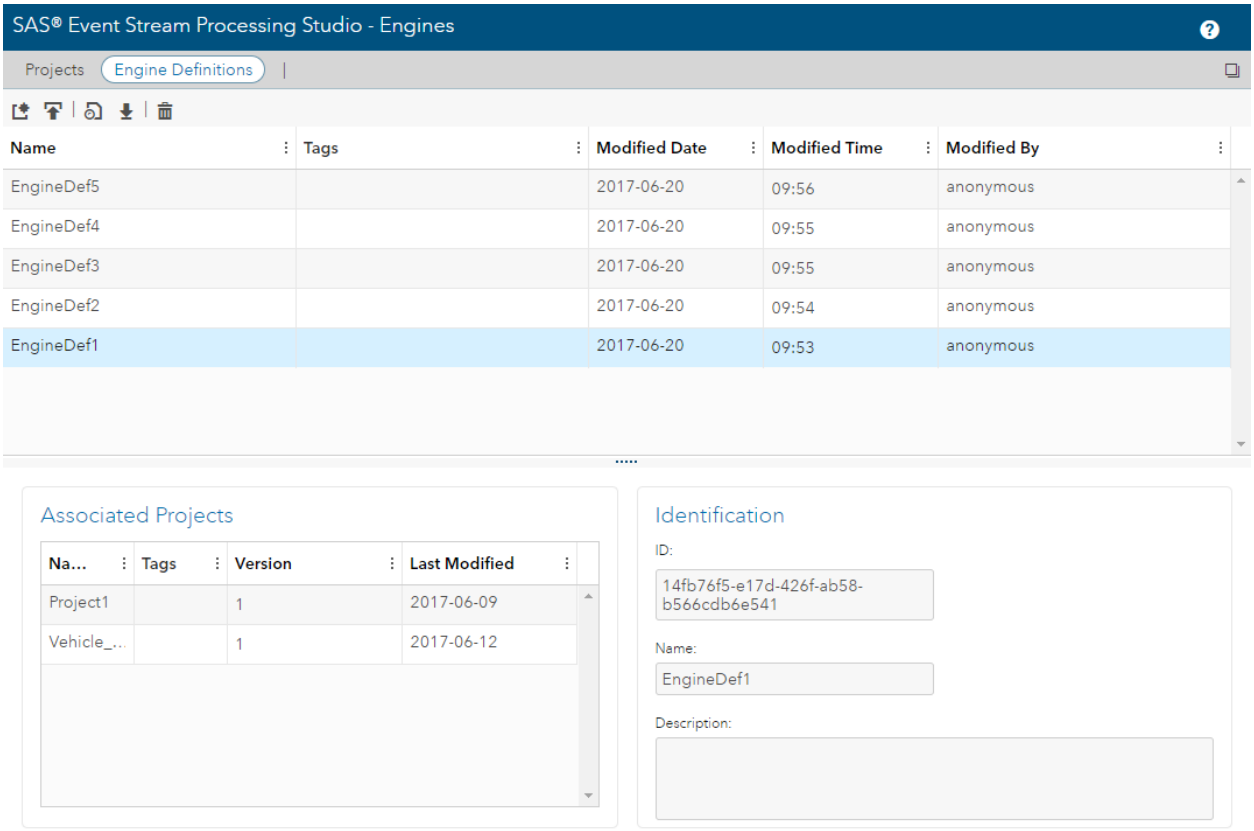
Working with Engine Definitions in SAS Event Stream Processing Studio

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 defined 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 associate one or more projects with each engine. You can view a list of the engine's associated projects in the **Associated Projects** pane.

The Engine Definitions Page



Creating Engine Definitions

To create a new engine definition:

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

The New Engine Definition window is displayed.

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 is displayed. This page consists of an **Identification** pane containing identifying information about the engine definition and an **Associated Projects** pane enabling you to associate the engine definition with one or more projects.

4 Review the information in the **Identification** pane. The **ID** field contains the engine definition's unique alphanumeric ID.

- 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 is displayed.

b In the **Project** field, select the project that you want your engine definition to be associated with and click **OK**.

The newly associated project is displayed in the **Associated Projects** pane.

6 Click  .

Uploading Engine Definitions

To upload an engine definition, do the following:

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

The Upload Engine Definition window is displayed.

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

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

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

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

6 In the **Version notes** field, enter any notes describing the version of 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 grid on the **Engine Definitions** page and click  .

Deleting Engine Definitions

To delete an engine definition, select the engine definition that you want to delete from the grid on the **Engine Definitions** page and click  . The engine definition is deleted from SAS Event Stream Processing Studio.

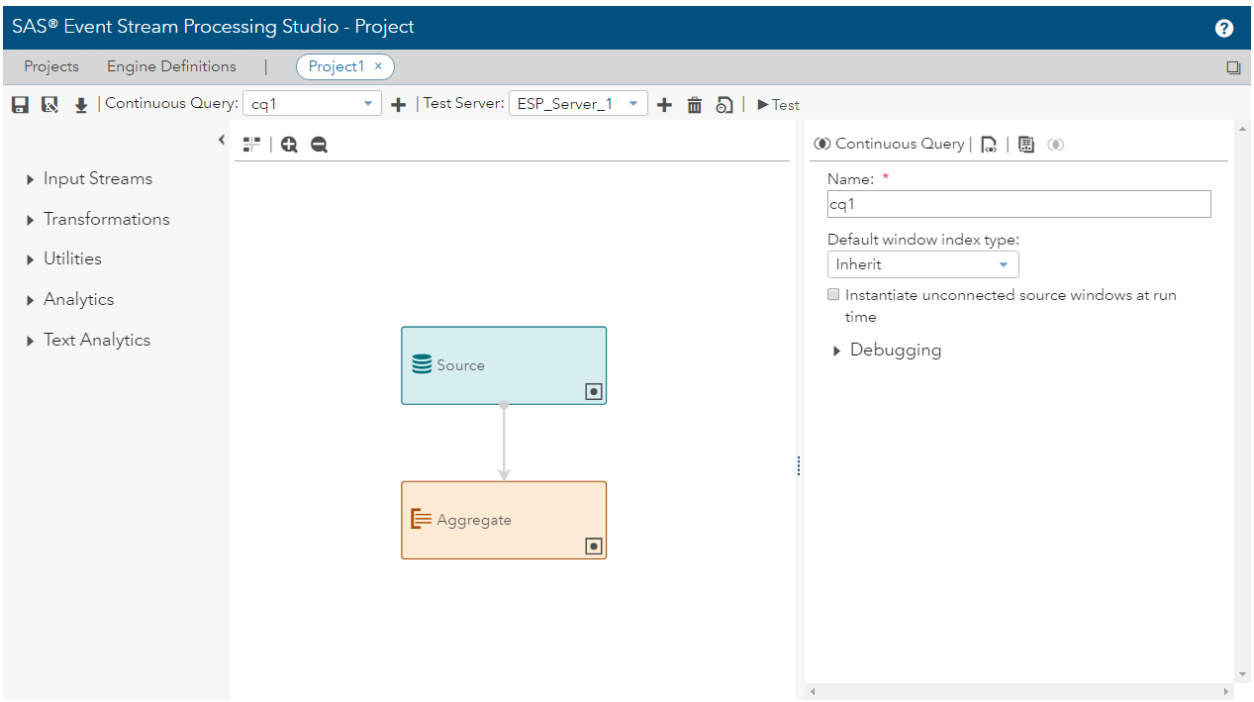
Using SAS Event Stream Processing Studio Modeler

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 is displayed 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.

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 20.

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.

Adding Windows

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

Note: You can also add windows to the currently displayed continuous query by double-clicking a window from the **Windows** pane on the left to the workspace.

The windows are grouped into the following categories:

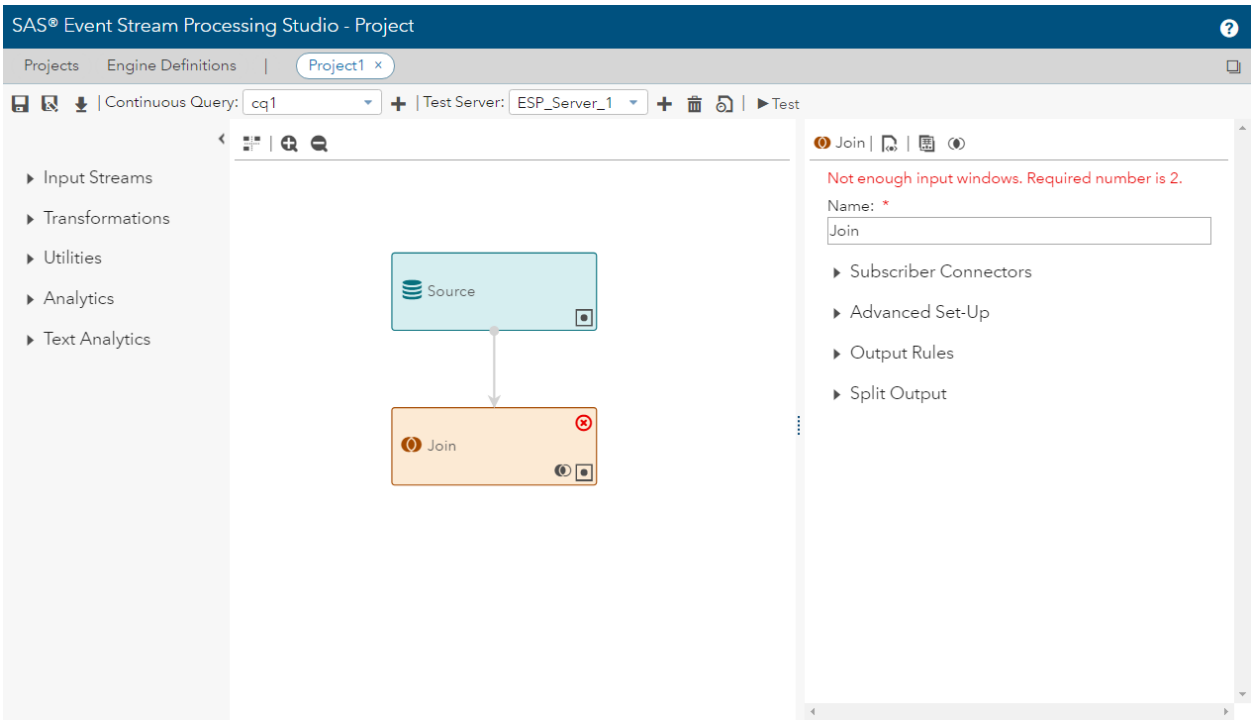
- Input Streams
- Transformations
- Utilities
- Analytics — To use these windows, you must have a license for SAS Event Stream Processing Analytics. If you do not have a license, you can construct a model, but you cannot execute the model on ESP server.

- Text Analytics — To use some of these windows, you must have a license for SAS Contextual Analytics (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 and test it, but you cannot execute the model on ESP server.

Note: The manually applied layout of windows on the workspace is retained after you close SAS Event Stream Processing Modeler. You can also apply an automatic layout to your model diagram by clicking  .

You must ensure that you enter valid properties for each window. Entering invalid window properties causes the window to display a  icon. Additionally, the right pane displaying the window's properties displays a warning message. For example, a model containing a source window and a join window can display the following warning message:

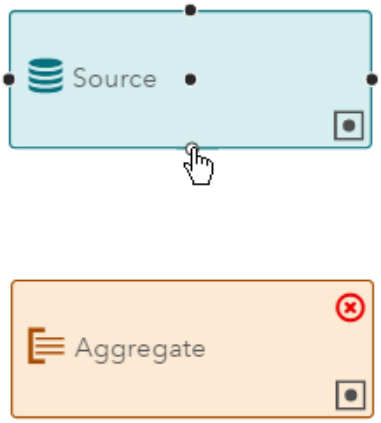
A Window Validation Warning Message



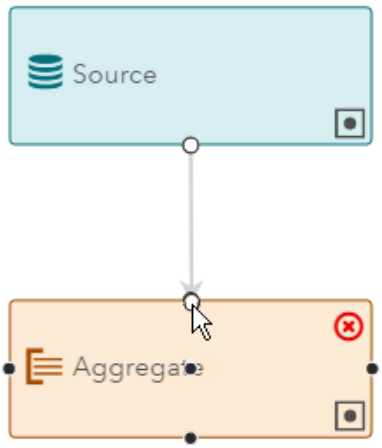
Connecting Windows

To connect a window to another window with an edge:

1 Position the cursor over an anchor point, so that the anchor point changes from black to white:



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

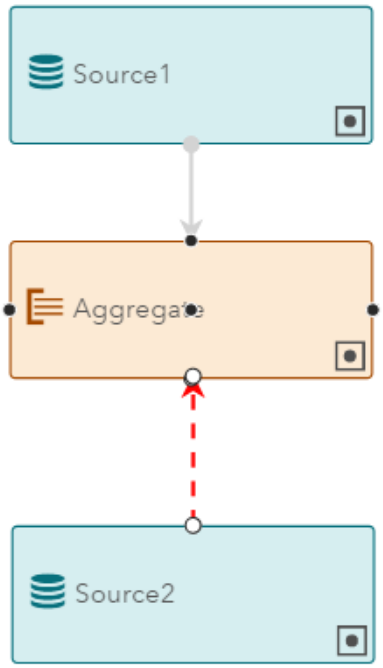


The edge automatically connects to the nearest anchor point.

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.

Edges that contain both multiple source windows and multiple target windows cannot be created in SAS Event Stream Processing Studio Modeler. For example, the following edge cannot be created: `<edge source="Source1 Source2" target="Join1 Join2"/>`.

An invalid edge is displayed as a red dashed line in SAS Event Stream Processing Studio Modeler. Here is an example:



Deleting Windows

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:

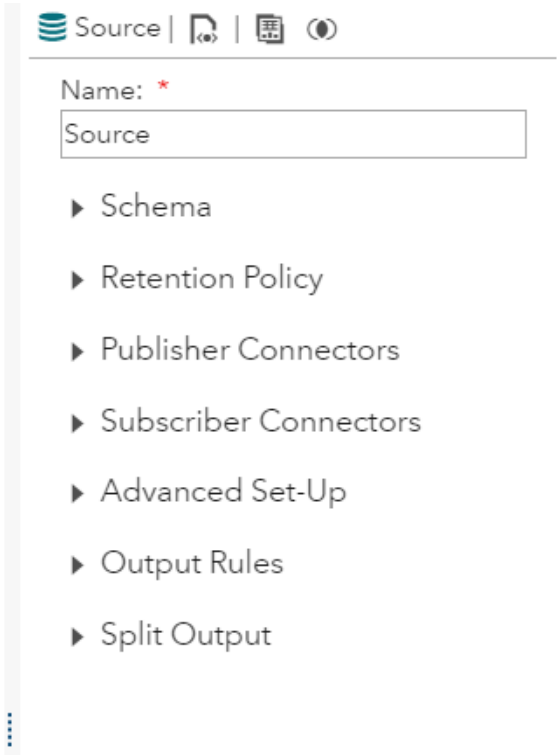
Icon	Description
	Indicates that the window's schema is invalid.
	Indicates that the window contains a fully stateful index. **Note:** If the window contains a non-stateful index, the Index state icon is not displayed.
	Indicates that the source window contains a connector. **Note:** Connector icons that are displayed on the left of a window indicate that the window contains a publisher connector. Connector icons that are displayed 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

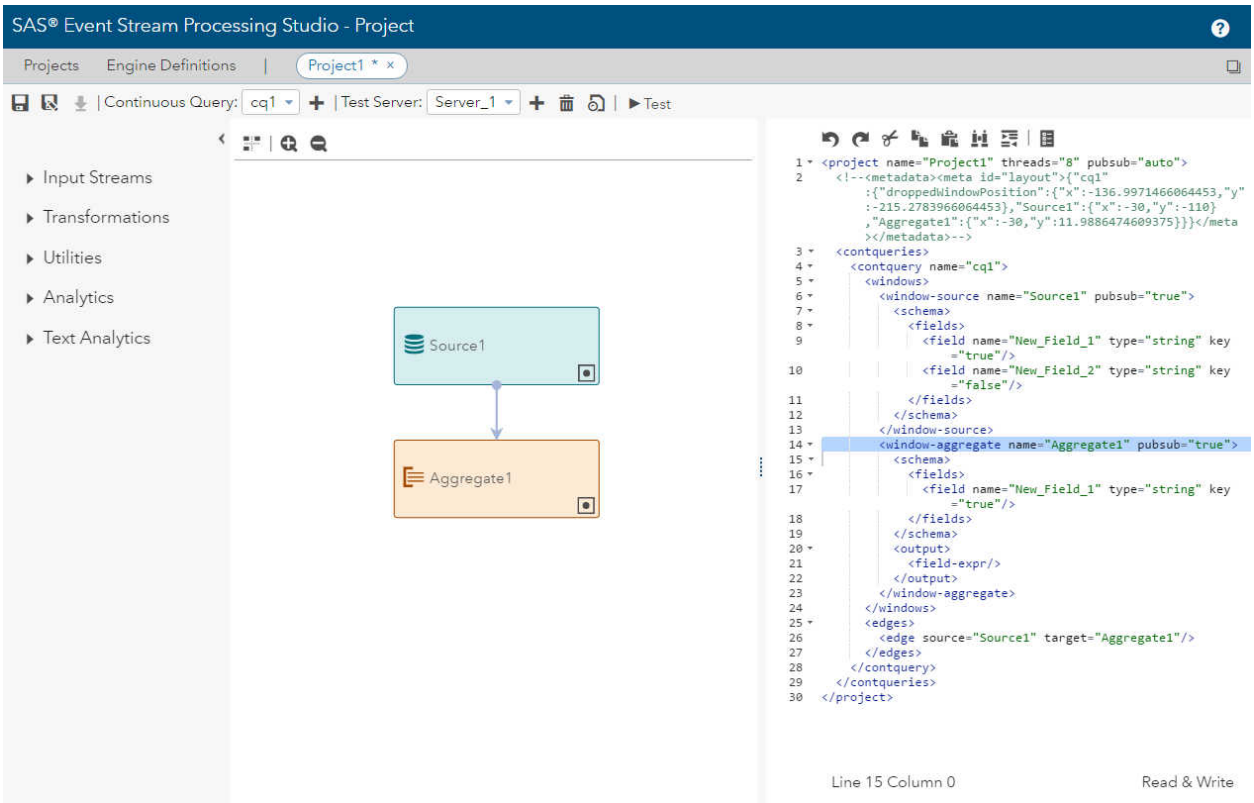
SAS Event Stream Processing Studio Modeler includes the XML Editor. You can use it as an alternative way of creating models, compared to the visual modeling capabilities in SAS Event Stream Processing Studio Modeler.

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. The workspace displays a snapshot of your model's XML code. Changes you make manually in the XML Editor are not always reflected in the workspace. Using the XML Editor to rename a window results in the old window being removed from the workspace and a new window being created in a default position. This invalidates your model. Any connections to or from the redundant window must 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



Select an element in your workspace to highlight the corresponding line of code in the XML Editor.

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 don't, 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 SAS Event Stream Processing schema.

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
	Removes the selected code from its original position.	Ctrl + X
	Copies the selected code to the clipboard.	Ctrl + C
	Pastes the code on the clipboard at the cursor's position.	Ctrl + V
	Searches for specific text. **Note:** Pressing Ctrl + F again replaces the text that you have searched for.	Ctrl + F
	Formats manually entered XML code.	not available
	Displays properties relevant to the selected window in the workspace.	not available

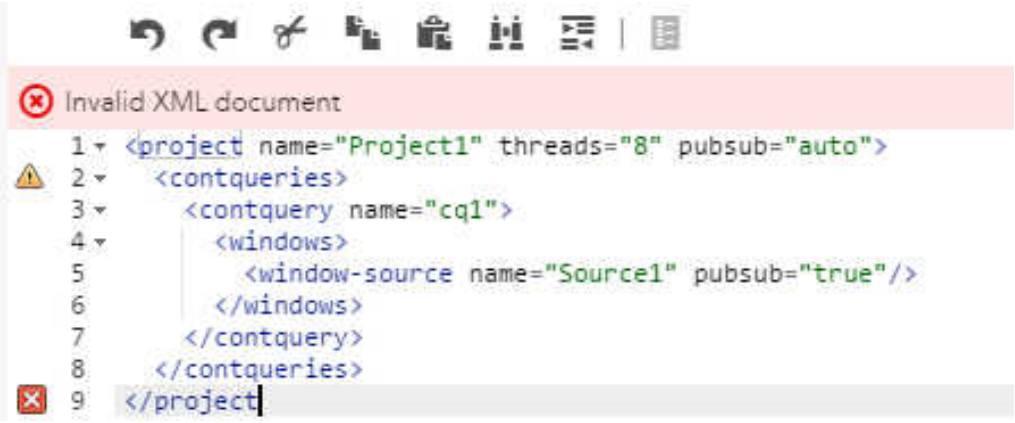
Validation

The XML Editor automatically validates the syntax of the code that you enter.

Note: The XML Editor does not perform any validation specific to SAS Event Stream Processing. That is, it does not validate your code against the SAS Event Stream Processing engine's schema.

If you enter invalid code, the XML Editor displays a warning:

Invalid XML Warning



Efficiency Tips

If windows in your model use the same fields, you can use the XML Editor to quickly copy and paste the fields between the windows. This is not possible when using the visual modeling capabilities in SAS Event Stream Processing Studio Modeler.

For example, if your model contains a source and an aggregate window that use the same schema fields, copy the contents of the source window's `<schema>` tag, as shown here:

Source Window Schema Fields to Be Copied to the Aggregate Window

:

Paste the schema fields into the aggregate window, as shown here:

Source Window Schema Fields That Have Been Copied to the Aggregate Window

```
<project name="Project1" threads="8" pubsub="auto">
  <contqueries>
    <contquery name="cq1">
      <windows>
        <window-source name="Source1" pubsub="true">
          <schema>
            <fields>
              <field name="New_Field_1" type="string" key="true"/>
              <field name="New_Field_2" type="string" key="false"/>
              <field name="New_Field_3" type="string" key="false"/>
            </fields>
          </schema>
        </window-source>
        <window-aggregate name="Aggregate1" pubsub="true"/>
        <schema>
            <fields>
              <field name="New_Field_1" type="string" key="true"/>
              <field name="New_Field_2" type="string" key="false"/>
              <field name="New_Field_3" type="string" key="false"/>
            </fields>
          </schema>
      </windows>
    </contquery>
  </contqueries>
</project>
```

Note: Alternatively, you can use SAS Event Stream Processing Studio Modeler to 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 press **OK**.

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

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

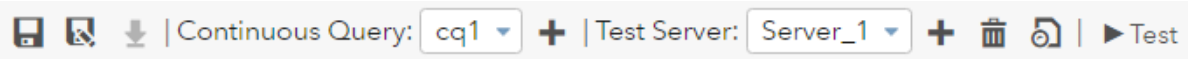
2. Configure the fields as required.

If the project is already open, and the right pane is not displaying the properties of the continuous query, click in the right pane.

To add a new continuous query to your model, click + next to the **Continuous Query** drop-down list on the toolbar.

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

   The project is displayed in a new tab.
3. Click ▶ Test.

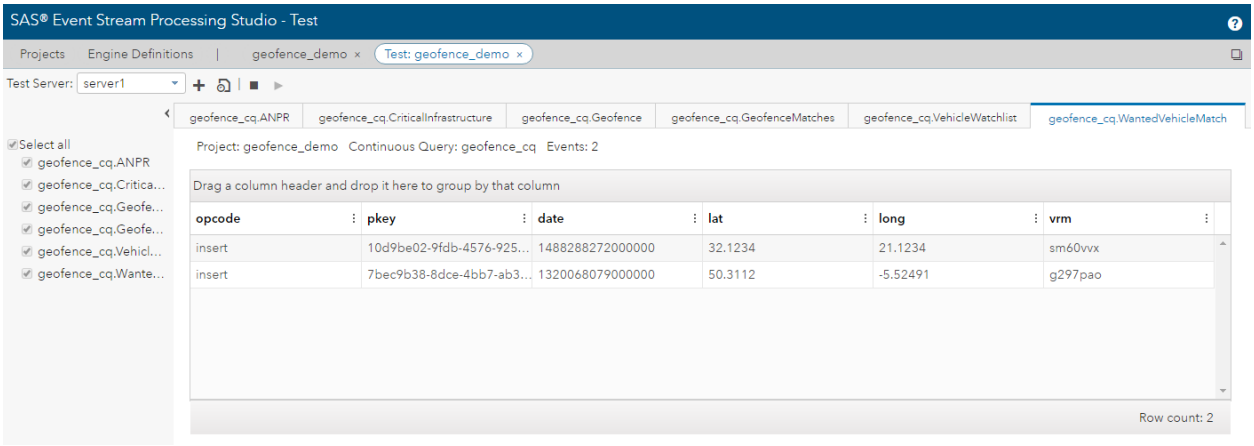
   A new tab called **Test** is displayed.
4. From the **Test Server** drop-down list, select a test server to perform the test on.

   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 is displayed 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 are displayed in their corresponding tab, as shown in the example image here. Only windows that have subscription enabled can display data. You can also view your test results by opening the output file that you specified in your subscriber connector properties.

Note: SAS Event Stream Processing Studio fetches data only for the tab that is currently displayed. When you click a different tab, SAS Event Stream Processing Studio then fetches data for that tab. If your data source contains a small number of events (for example, the data source is a small CSV file rather than a stream of data), and you view a tab after all the events that are relevant for the corresponding window have already passed, the tab is empty. That is, there can be situations where all the data for a specific tab has passed before you view that tab. If the data has already passed through before you have switched tab, run the test again and select a different tab.



7 To stop the test, click ■ and close the tab.

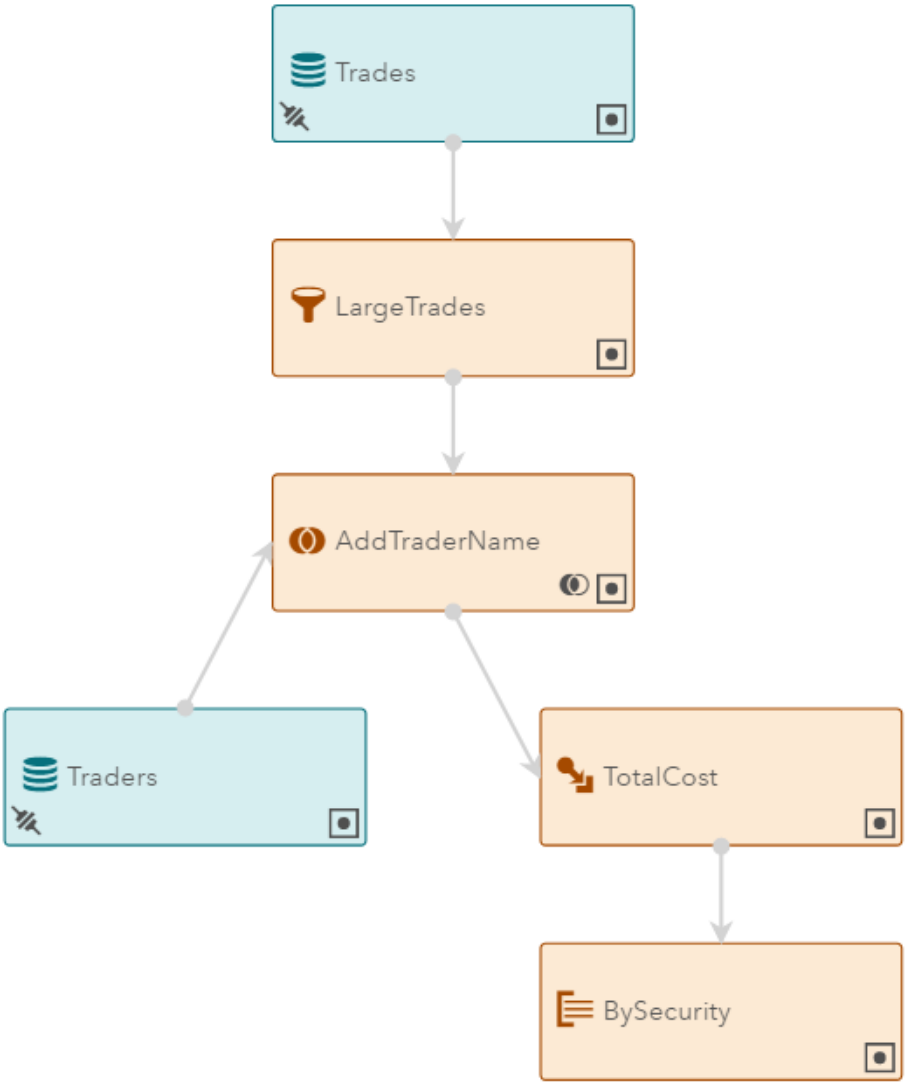
Note: If the test fails and the error message does not provide enough information about what caused the failure, then SAS Event Stream Processing engine log files might provide additional information. For more information, see "Logging" in SAS Event Stream Processing: Troubleshooting.

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 .

Your project is created and SAS Event Stream Processing Studio Modeler is displayed. However, if you do not currently have any test servers configured, the Event Stream Processing Server window is displayed first.

2 If the Event Stream Processing Server window is displayed, do the following:

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 of the test server.

c In the **Admin Port** field, enter the test server's administration port number.

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

e Click **OK**.

3 Click  .

The New Project window is displayed. This window enables you to record general information about the project.

4 In the New Project window, do the following:

a In the **Project 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**.

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

6 Click  .

The right pane displays the project's properties.

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

8 In the **Thread pool size** field, change the thread pool size to `4`.

9 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.

You will configure this window to receive events about stock market securities transactions.

10 In the **Name** field, change the default name to `Trades`.

11 Specify a schema for the Trades window:

a Expand **Schema**.

b Click  .

The Edit Output Schema window is displayed.

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:

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

d Click **OK**.

12 You will 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 Expand **Publisher Connectors**.

b Click  .

The Publisher connectors window is displayed.

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

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/trades_xml/trades.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 In the **dateformat** field, enter **`%d/%b/%Y:%H:%M:%S`**.

g Click **OK**.

13 Collapse **Publisher Connectors**.

14 Expand **Advanced Set-Up** and set the **Index** drop-down list to **pi_RBTREE**.

15 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.

You will configure this window to receive information about stock market traders.

16 In the **Name** field, change the default name to **`Traders`**.

17 Specify a schema for the Traders window:

a Expand **Schema**.

b Click  .

The Edit Output Schema window is displayed.

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:

Name	Type	Key
ID	Int64	Y
name	String	N

d Click **OK**.

18 You will 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 Expand **Publisher Connectors**.

b Click  .

The Publisher connectors window is displayed.

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

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/trades_xml/traders.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 Click **OK**.

19 Expand **Transformations** on the **Windows** pane on the left and drag a filter window to the workspace.

You will 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.

20 Click the newly created filter window on the workspace.

The right pane displays the filter window's properties.

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

22 Expand **Output Fields**, and enter `quantity >= 100` in the **Expression** field.

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

a Position the mouse pointer over an anchor point in 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 an anchor point in the LargeTrades window.

The LargeTrades window now accepts trades from the Trades window.

24 Expand **Transformations** on the **Windows** pane on the left and drag a join window to the workspace.

You will configure this window to match large trades with the traders who made those trades.

25 Click the newly created join window on the workspace.

The right pane displays the join window's properties.

26 In the **Name** field, change the default name to **`AddTraderName`**.

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

The AddTraderName window now accepts trades from the LargeTrades window.

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

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

29 Click the AddTraderName window on the workspace.

The right pane displays the join window's properties.

30 Expand **Join Criteria** 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.

31 Set the **Join Type** drop-down field to **LeftOuter**.

32 Configure join conditions:

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

b Click the cell in the Left Fields column, and select **traderID**.

c Click the cell in the Right Fields column, and select **ID**.

33 Collapse **Join Criteria**.

34 Specify a schema for the AddTraderName window:

a Expand **Output Schema**.

b Click .

The Edit Output Schema — Non Key Fields window is displayed. Use this window to configure the fields as shown in the following table. After you add a row, click again to add the next row.

Name	Window	Field
security	LargeTrades	security
quantity	LargeTrades	quantity
price	LargeTrades	price
traderID	LargeTrades	traderID
time	LargeTrades	time
name	Traders	name

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 press **OK**.

35 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.

You will configure this window to compute the total cost of the large trades.

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

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

The TotalCost window now accepts trades from the AddTraderName window.

38 Click the TotalCost window to display the compute window's properties in the right pane again.

39 Specify a schema for the TotalCost window:

a Expand **Output Schema**.

b Click  .

The Edit Output Schema window is displayed.

c Click  to add a row to the schema table.

Enter the following value:

Field Name	Type	Expression	Key
tradeID	String	(not applicable)	Y
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	N

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 press **OK**.

d Click **OK**.

40 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.

You will configure this window to compute the total cost of the large trades.

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

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

The BySecurity window now accepts trades from the TotalCost window.

43 Click the BySecurity window to display the aggregate window's properties in the right pane again.

44 Specify a schema for the BySecurity window:

a Expand **Output Schema**.

b Click  .

The Edit Output Schema window is displayed.

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

Key	Name	Type	Aggregate Function	Fields / Parameters
Y	security	String	(not applicable)	(not applicable)
N	quantityTotal	Double	ESP_aSum(fieldName)	quantity
N	costTotal	Double	ESP_aSum(fieldName)	totalCost

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 press **OK**.

d Click **OK**.

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

46 Click ▶ Test .

A new page called **trades_proj — Test 1** is displayed.

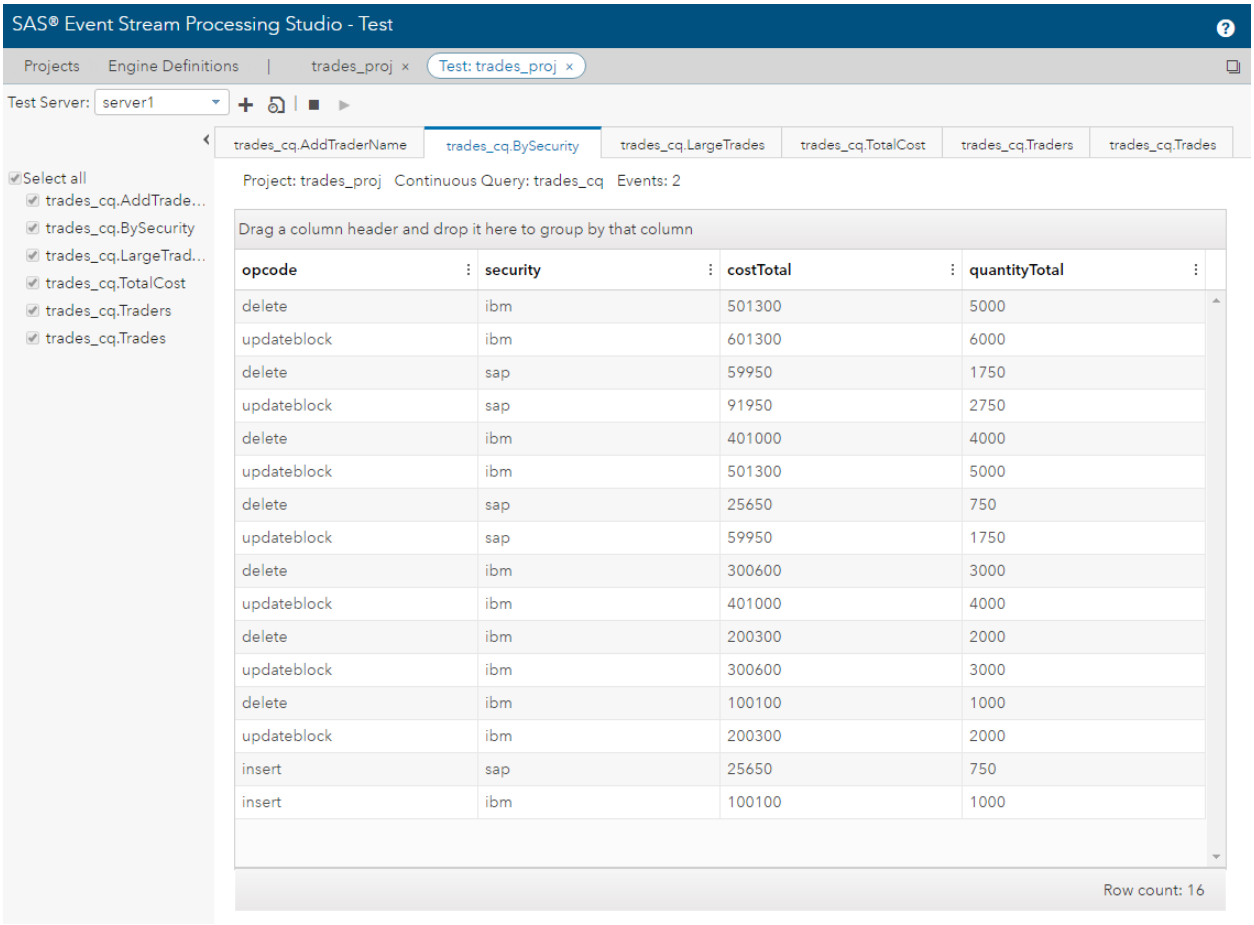
47 In the **Test Server** drop-down list, select the test server on which you want to test the model.

48 Click ▶.

The results for each window are displayed on separate tabs:

- the **Trades** tab lists the stock market securities transactions
- the **Traders** tab lists the traders
- the **LargeTrades** tab lists the large trades
- the **AddTraderName** tab lists the large trades and includes an additional column that shows trader names
- the **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 **BySecurity** tab shows all the inserts, deletes, and updateblocks for the large trades, as illustrated in the following figure. The newest event is shown at the top of the table. Rows 2 and 4 show the total cost of transactions for each security: 601300 for IBM and 91950 for SAP

Results for the BySecurity Window



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.

49 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.

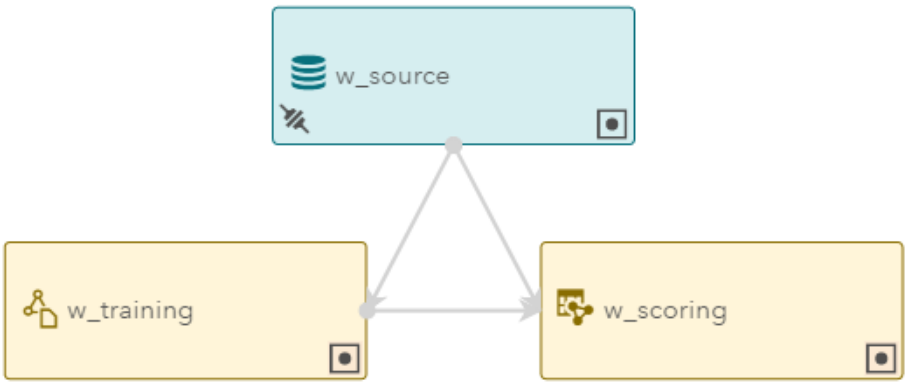
Note: To successfully complete this example, you must have access to SAS Event Stream Processing Analytics — a separately orderable and licensed package that enables the use of advanced analytics and machine learning techniques. Your deployment must also contain the esp_sa_plugin library.

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 .

   Your project is created and SAS Event Stream Processing Modeler is displayed. However, if you do not currently have any test servers configured, the Event Stream Processing Server window is displayed first.

2 If the Event Stream Processing Server window is displayed, do the following:

   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 of the test server.

   c In the **Admin Port** field, enter the test server's administration port number.

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

   e Click **OK**.

3 Click .

   The New Project window is displayed.

4 In the New Project window, do the following:

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

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

   c Click **OK**.

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

6 Expand **Debugging**. In the **Trace in server log** field, enter `w_scoring w_training`.

7 Click .

The right pane displays the project's properties.

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

b In the **Thread pool size** field, change the thread pool size to `1`.

c Select the **When source window output is split, then rejoined in a later window, rejoin event blocks with the same transaction ID** check box.

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

10 Specify a schema for the Events_Source window:

a Expand **Schema**.

b Click  .

The Edit Output Schema window is displayed.

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

Enter the following values in the rows:

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

d Click **OK**.

11 The Events_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 Expand **Publisher Connectors** and click  .

The Publisher connectors window is displayed.

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

c 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.

d In the **fstype** drop-down list, select **csv**.

e In the **transactional** drop-down list, select **true**.

f In the **blocksize** field, enter `1`.

g Click **OK**.

h Collapse **Publisher Connectors**.

12 Expand **Advanced Set-Up**.

13 Select the **Only accept insert events** check box.

14 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.

15 In the right pane, in the **Name** field, change the default name to `w_training`.

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

17 Expand **KMEANS**.

a Expand **Properties**.

b In the **nClusters** field, enter `2` to specify the number of clusters.

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, enter `0.8` to specify the damping factor for old data points.

e In the **fadeOutFactor** field, enter `0.05` to specify the factor for determining whether an existing cluster is fading out.

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, enter `0.01` to specify the factor for the disturbance when splitting a cluster.

g In the **nInit** field, enter `50` to specify the number of data events used during initialization.

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

i In the **commitInterval** field, enter `25` to specify the number of timestamps to elapse before committing a model to downstream scoring.

j Collapse **Properties**.

k Expand **Input Map**.

l In the **inputs** field, enter `x_c,y_c` to specify the variable names to use in clustering.

18 Connect the w_source window to the w_training window with an edge:

a Position the mouse pointer over an anchor point in 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 an anchor point in the w_training window.

The w_training window now accepts events from the w_source window.

19 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.

20 In the right pane, in the **Name** field, change the default name to `w_scoring`.

21 Specify a schema for the w_scoring window:

a Expand **Output Schema**.

b Click .

The Edit Output Schema window is displayed.

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:

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

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 press **OK**.

22 Click **OK**.

23 Expand **Set Up**.

a Select the **KMEANS** box.

b Expand **KMEANS**.

c Expand **Input Map**.

d Specify the variables to use in clustering. In the **inputs** field, enter **x_c,y_c**.

e Expand **Output Map**.

f Specify the output variable name in the output schema that stores the cluster label. In the **labelOut** drop-down list, select **seg**.

g Specify the output variable name in the output schema that stores the distance to the nearest cluster. In the **minDistanceOut** drop-down list, select **min_dist**.

h 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** drop-down list, select **model_id**.

24 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.

25 Connect the w_training window to the w_scoring window with an edge.

26 Click  to apply an automatic layout to your model.

27 Click  .

28 Click ▸ Test .

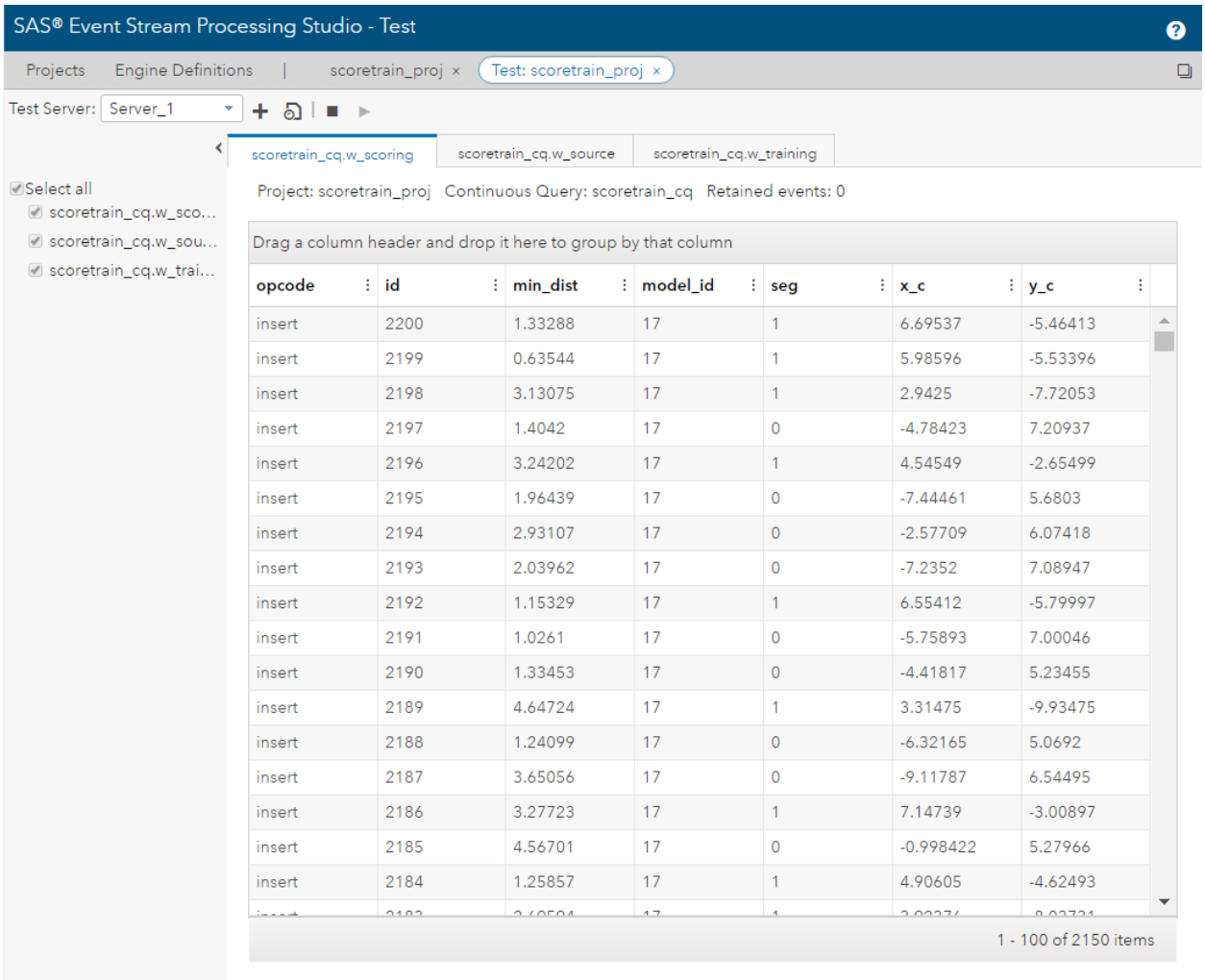
A new page called **Test scoretrain_proj** is displayed.

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

30 Click  .

The results for each window are displayed 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, as shown in the following figure:



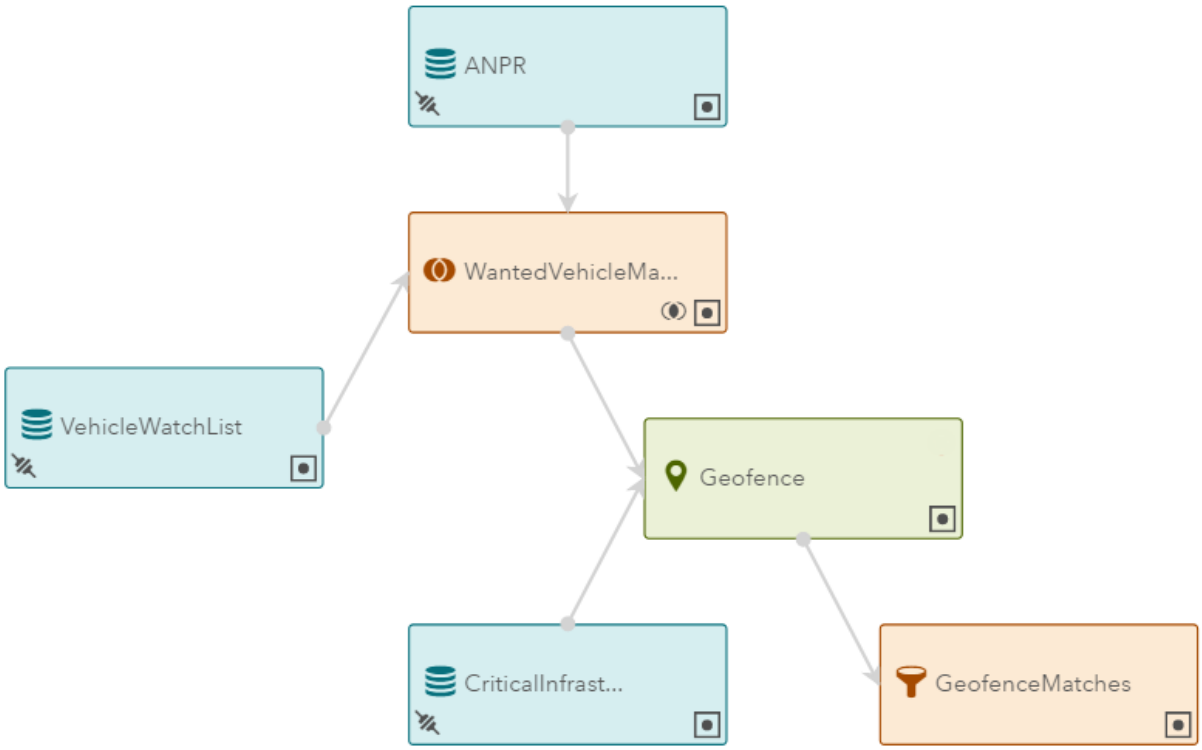
31 To stop the test, click  .

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 from the following location: `ftp://ftp.sas.com/techsup/download/esp/geofence2_xml.tar.gz`.

1 Download the geofence2_xml.tar.gz file from `ftp://ftp.sas.com/techsup/download/esp/geofence2_xml.tar.gz`.

2 Navigate to the geofence2_xml.tar.gz file that you downloaded and extract its contents.

Note: It is recommended that you extract the files into a folder called geofence2_xml. Make a note of the location that you extracted the files to.

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

Your project is created and SAS Event Stream Processing Studio Modeler is displayed. However, if you do not currently have any ESP servers configured, the Event Stream Processing Server window is displayed first.

4 If the Event Stream Processing Server window is displayed, do the following:

   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 of the test server.

   c In the **Admin Port** field, enter the test server's administration port number.

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

   e Click **OK**.

5 Click  .

   The New Project window is displayed.

6 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**.

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

8 Click  .

   The right pane displays the project's properties.

9 In the **Name** field, enter `geofence`.

10 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.

11 In the **Name** field, change the default name to `ANPR`.

12 Specify a schema for the ANPR window:

   a Expand **Schema**.

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

   c Click  .

   The Edit Output Schema window is displayed.

   d 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:

Name	Type	Key
vrm	String	N

Name	Type	Key
lat	Double	N
long	Double	N
date	TimeStamp	N
pkey	String	Y

e Click **OK**.

f Collapse **Schema**.

13 Configure the ANPR window to only accept input events:

a Expand **Advanced Set-Up**.

b Select the **Only accept insert events** check box.

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

a Expand **Publisher Connectors** and click .

The Publisher connectors window is displayed.

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

c 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.

d In the **fstype** drop-down list, select **csv**.

e In the **dateformat** field, enter `%Y-%m-%d %H:%M:%S`.

f In the **header** file, enter `1`.

g In the **ignorecsvparseerrors** drop-down list, select **true**.

h Click **OK**.

15 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.

16 In the **Name** field, change the default name to `VehicleWatchList`.

17 Specify a schema for the VehicleWatchList window:

a Expand **Schema**.

b Click .

The Edit Output Schema window is displayed.

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

Name	Type	Key
vrm	String	Y

d Click **OK**.

e Collapse **Schema**.

18 The VehicleWatchList window will stream a list of wanted vehicles from a file called wantedvehicle.csv that contains example data. To add a connector to this CSV file:

a Expand **Publisher Connectors** and click .

The Publisher connectors window is displayed.

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

c 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.

d In the **fstype** drop-down list, select **csv**.

e In the **header** file, enter `1`.

f Click **OK**.

19 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.

20 In the **Name** field, change the default name to `WantedVehicleMatch`.

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

The WantedVehicleMatch window now accepts values from the ANPR window.

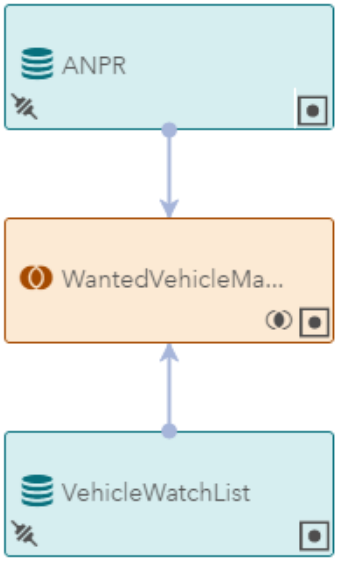
a Position the mouse pointer over an anchor point in the ANPR window so that the anchor point changes from black to white.

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

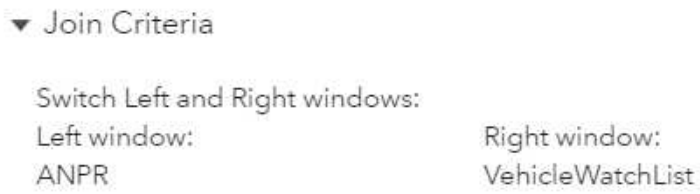
22 Connect the VehicleWatchlist window to the WantedVehicleMatch window with an edge.

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 15.



23 Expand **Join Criteria** and 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.



24 Click the WantedVehicleMatch window in the workspace.

The right pane displays the join window's properties.

25 Configure the WantedVehicleMatch window's join criteria:

a Expand **Join Criteria**.

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

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

d Click the cell in the Left Fields column, and select **vrm**.

e Click the cell in the Right Fields column, and select **vrm**.

f Collapse **Join Criteria**.

26 Specify an output schema for the WantedVehicleMatch window:

a Expand **Output Schema**.

b In the **Output select fields** field, click to add a row to the schema table.

The Edit Output Schema — Non Key Fields window is displayed. Use this window to configure the fields as shown in the following table. After you add a row, click again to add the next row.

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

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 press **OK**.

c Click **OK**.

27 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.

28 In the **Name** field, change the default name to `CriticalInfrastructure`.

29 Specify a schema for the CriticalInfrastructure window:

a Expand **Schema**.

b Click .

The Edit Output Schema window is displayed.

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:

Name	Type	Key
name	String	Y
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

Name	Type	Key
notes	String	N

d Click **OK**.

e Collapse **Schema**.

30 The CriticalInfrastucture window will stream a list of sites containing critical infrastructure from a file called infrastructure.csv that contains example data. To add a connector to this CSV file:

a Expand **Publisher Connectors** and click .

The Publisher connectors window is displayed.

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

c 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.

d In the **fstype** drop-down list, select **csv**.

e In the **header** file, enter `1`.

f Select **true** from the **ignorecsvparseerrors** drop-down list.

g Click **OK**.

h Collapse **Publisher Connectors**.

31 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.

32 In the **Name** field, change the default name to `Geofence`.

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

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

35 Click the Geofence window in the workspace.

The right pane displays the join window's properties.

36 Configure the Geofence window's positional settings:

a Expand **Positions**.

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

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

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

e Collapse **Positions**.

37 Configure the Geofence window's geometric settings:

a Expand **Geometries**.

b In the **Circle data - Longitude coordinate** drop-down list, select **long**.

c In the **Circle data - Latitude coordinate** drop-down list, select **lat**.

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

e Collapse **Geometries**.

38 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**.

39 Configure the Geofence window's output properties:

a Expand **Output**.

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

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

d Collapse **Output**.

40 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.

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

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

43 Click  to apply an automatic layout to your model.

44 Click the GeofenceMatches window to specify its output fields:

a Expand **Output Fields**.

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

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

46 Click .

A new page called **geofence_demo — test 1** is displayed.

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

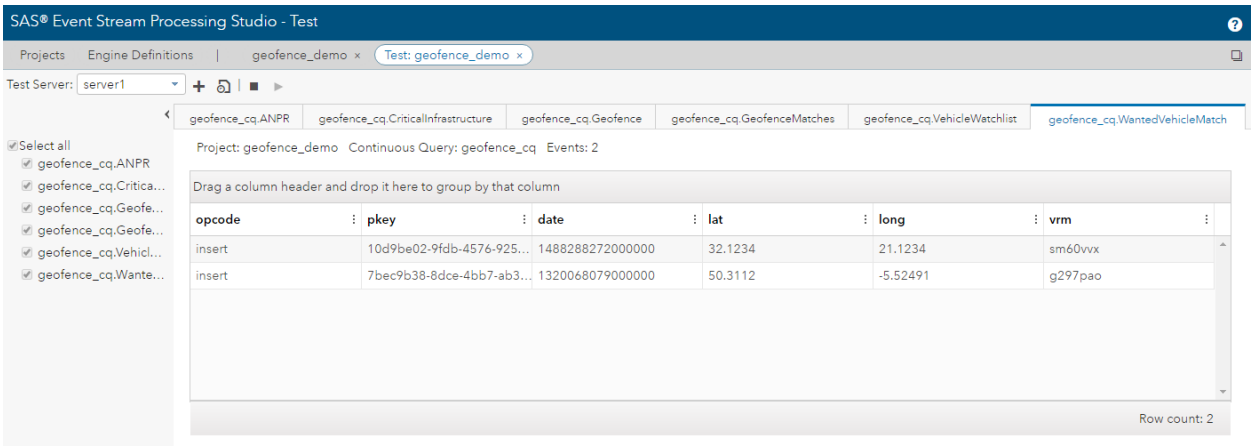
48 Click  .

The results for each window are displayed on separate tabs:

- the **ANPR** tab lists all vehicles within close proximity of critical infrastructure sites
- the **VehicleWatchlist** tab lists all vehicles on the vehicle watch list
- the **WantedVehicleMatch** tab lists the wanted vehicles found within close proximity of critical infrastructure sites, as shown in the image here
- the **Geofence** tab lists the geofencing information relating to the matched vehicles
- the **CriticalInfrastructure** tab lists sites containing critical infrastructure

- the **GeofenceMatches** tab shows wanted vehicles found within close proximity of critical infrastructure sites

Results for the WantedVehicleMatch Window



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.

49 To stop the test, click ■.

The project stops and then unloads from the ESP 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. You can also import existing modules into SAS Event Stream Processing Studio from SAS Model Manager.

Creating SAS Micro Analytic Service Modules

To create a new module:

1 Open your project and click .

2 Expand **MAS Modules**.

3 Click .

The MAS Module window is displayed.

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 **Code source** field, select **Embedded code** to enter your own code or select **External file** to use code located in an external file.

8 If you selected **Embedded code** in the **Code source** field, enter your code in the **Embedded code** field. If you selected **External file** in the **Code source** field, enter the file path to the external file in the **External file** field.

9 In the **Function names** field, enter your module's function names.

10 Click **OK**.

The module you created is displayed in the MAS Module grid.

Uploading SAS Micro Analytic Service Modules

To upload an existing module:

1 Open your project and click  .

2 Expand **MAS Modules**.

3 Click  .

The Import A MAS Module From A SAS Model Manager ZIP File window is displayed.

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 MAS Module From A SAS Model Manager ZIP File window reloads to display information on the module's roles.

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

7 Click **OK.**

The module you uploaded is displayed in the MAS Module grid.

Deleting SAS Micro Analytic Service Modules

To delete a module, select the module that you want to delete from the MAS Module grid and click  .

The module is deleted from the MAS Module grid.

Importing SAS Micro Analytic Service Modules Created in SAS Model Manager Directly into Your Model

You can import content created in SAS Model Manager directly into a specific procedural window in your model. The module is uploaded and then referenced from the procedural window's input handler.

To import a SAS Micro Analytic Service module directly into a procedural window:

1 Open the project that you want to import the module to.

2 Click the relevant procedural window.

3 Expand **Input Handler Functions**.

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

5 Click  .

The Input Handler Functions window is displayed.

6 In the **Handler type** field, to import a module, select **Import a MAS module from a SAS Model Manager ZIP file**.

7 Click **Choose File**.

8 Browse 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 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 Handler Functions window is displayed.

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.

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 project that you want to import the SAS Model Manager content to.

2 Click the relevant procedural window.

The right pane displays the properties of the procedural window.

3 Expand **Input Handler Functions**.

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

5 Click  .

The Input Handler Functions window is displayed.

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.

- Micro Analytic Service (MAS) – enables you to reference a SAS Micro Analytic Service module

The Input Handler Functions 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.

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 Expand **Input Handler Functions**.

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

5 Click  .

The Input Handler Functions window is displayed.

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

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 Handler Functions window is displayed.

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.

Analytic Store files

Importing analytic store (ASTORE) files created in SAS Model Manager is not fully supported in SAS Event Stream Processing Studio 4.3. To import an ASTORE file from SAS Model Manager into SAS Event Stream Processing Studio, you must manually extract the ASTORE file from the SAS Model Manager ZIP file and copy this file onto the ESP server. You must then register the imported ASTORE file in the relevant score window in your model.