SAS® Event Stream Processing 6.2: Using SAS® Event Stream Processing Studio

SAS Event Stream Processing Studio Overview

What Is SAS Event Stream Processing Studio

SAS Event Stream Processing Studio is a web-based client that enables you to create, edit, upload, publish, 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

- A supported web browser has been installed. For information about how to view the browsers that are supported, see “Accessing SAS Event Stream Processing Studio” on page 2.
- Your screen has a minimum resolution of 1,280 x 1,024.

- JavaScript has been enabled in your browser.

Accessing 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: If Transport Layer Security (TLS) is enabled on the SAS Event Stream Processing server in your deployment, you must access SAS Event Stream Processing Studio by using the HTTPS protocol in the browser's URL. Failure to use the HTTPS protocol can cause SAS Event Stream Processing Studio to not operate as expected.

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 the application.

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

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

To access information about the browsers that are supported, click the user icon in the top right corner in the application and then click **About**. Click **Supported Browsers and Platforms** to view the browsers that are supported. SAS Event Stream Processing Studio requires the use of cookies to maintain the session state.

Starting the Event Stream Processing Server

If you are using a non-Kubernetes environment, you must configure an ESP server before you can successfully run a model. For more information, see "Managing ESP Servers in SAS Event Stream Processing Studio" on page 37. If you are using a Kubernetes environment, an ESP server is created for you on-demand and you do not need to manually create one. For more information, see "Working with a Kubernetes Cluster in SAS Event Stream Processing Studio" on page 40.

To start the ESP server on a UNIX system, run the following command:

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

On Windows systems, run the following command:

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

-http port specifies the port for the HTTP REST API. You can check the terminal log to confirm that you have instantiated the ESP server successfully. In addition, the terminal log displays the port specified here.

-pubsub port specifies the port for publish and subscribe actions.

In these examples, **`$DFESP_HOME`** is the installation directory on UNIX systems and **`%DFESP_HOME%`** is the installation directory on Windows systems.

Note: On Windows systems, if your installation directory contains a space, you must enclose this command in quotation marks, as shown here:

```
"%DFESP_HOME%\bin\dfesp_xml_server" -pubsub 7003 -http 1234
```

For information about the ESP server, see "Setting Up and Using the ESP Server" in *SAS Event Stream Processing: Using the ESP Server*.

Understanding the User Interface

Pages

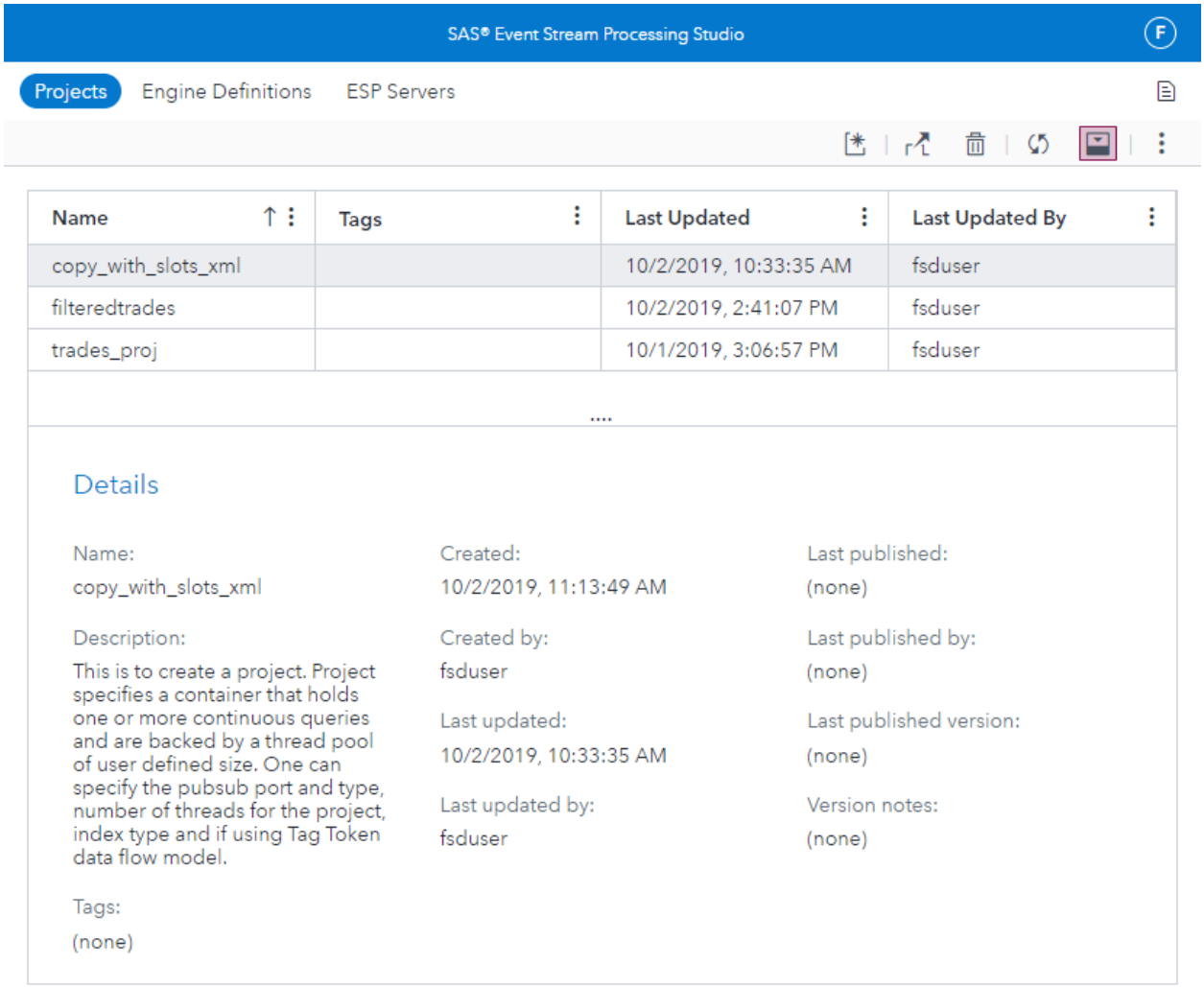
A *page* is the highest level container in the user interface. All other user interface elements are contained within a page.

SAS Event Stream Processing Studio contains the following main pages:

- the **Projects** page enables you to create, edit, upload, download, or delete the projects that contain your models
- the **Engine Definitions** page enables you to create, edit, upload, download, or delete engine definitions
- the **ESP Servers** page enables you to create, edit, upload, download, or delete ESP servers

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

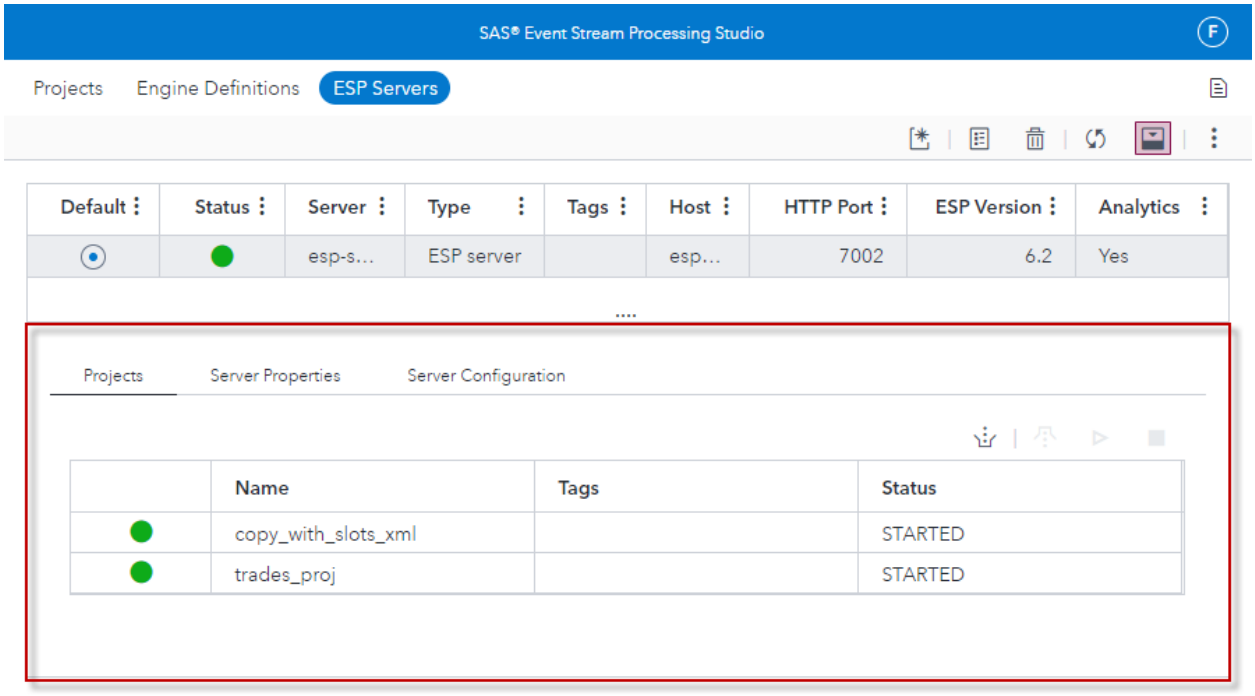
Figure 1 *The Projects Page Displaying Active Test Projects*



Panes

Panes that enable you to view different types of information within the same page. The following figure displays the **ESP Servers** page, which contains a bottom pane. In this example, the pane contains further information about the ESP server selected.

Figure 2 *Example of a Horizontal Pane*

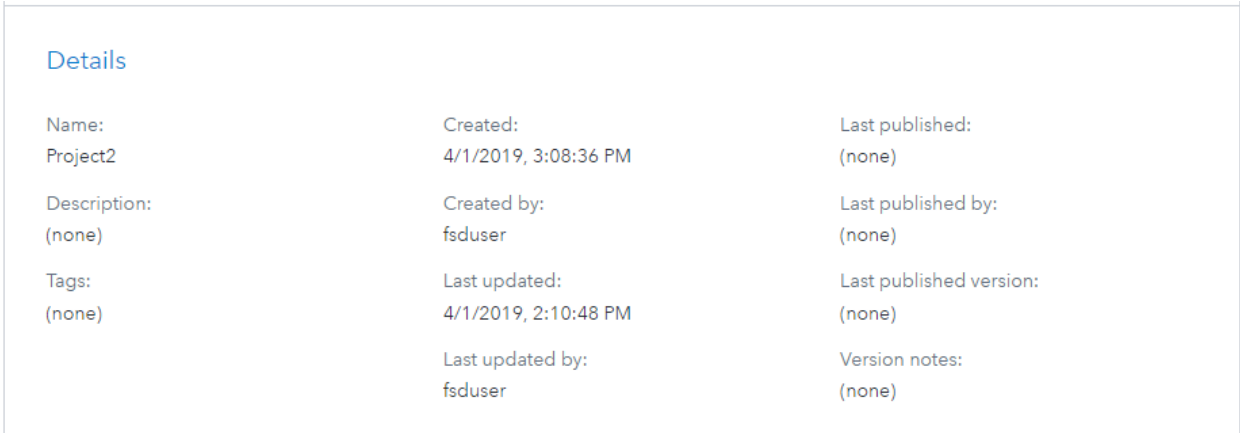


To resize a pane, drag a border in the appropriate direction. To resize a horizontal pane, drag a border upward or downward. To resize a vertical pane, drag the border left or right.

Tiles

A *tile* is a self-contained block of information that resides within a pane or sometimes directly on a page.

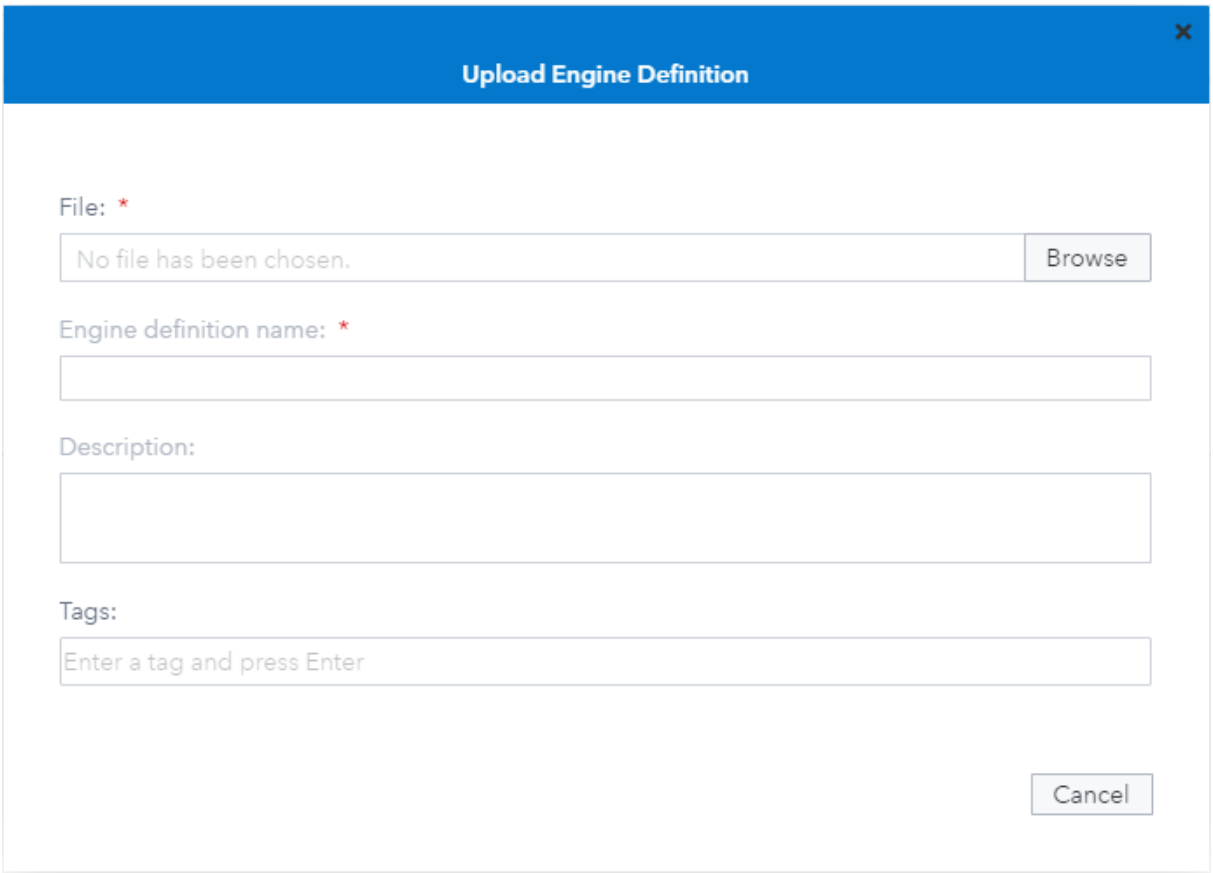
Figure 3 *Example of a Single Tile inside a Horizontal Pane*



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.

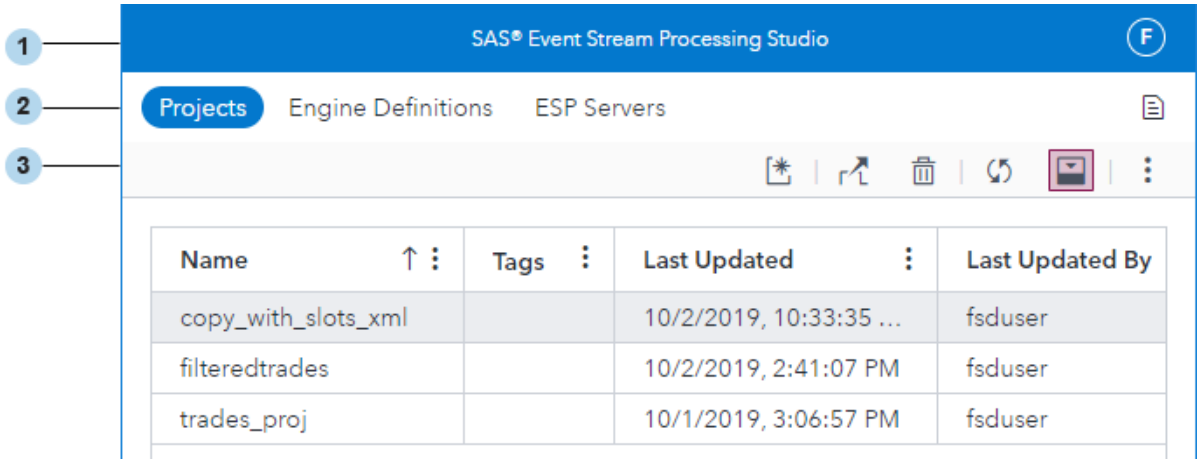
Figure 4 *The Upload Engine Definition Window*



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

Figure 5 *The Application Toolbars*



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

Item Number	Description	Name
1	Application bar	■ Displays the product name. ■ Displays the first character of your display name or user ID. Clicking on this character does the following: □ Shows your display name or your user ID in full. If you have not set a display name, your user ID is displayed by default. 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. □ Provides access to Help and product information □ Enables you to sign out of SAS Event Stream Processing Studio (if applicable)
2	Menu bar	■ Provides access to the main SAS Event Stream Processing Studio pages: **Projects**, **Engine Definitions**, and **ESP Servers** ■ Provides access to each project, project version, model test, or engine definition that is currently open. The navigation overflow menu button displays the total number of these pages that are currently open. For example, if one page is open, the following icon appears: 1. If there are no pages currently open, the following icon appears: .
3	Toolbar	■ Includes buttons or tabs associated with the open item ■ Enables you to perform actions associated with a project. For example, you can download a project to your computer by clicking ⋮ on the toolbar and selecting **Download**.

Sorting and Filtering

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 18. For more information about the XML Editor, see “Using the XML Editor” on page 28.

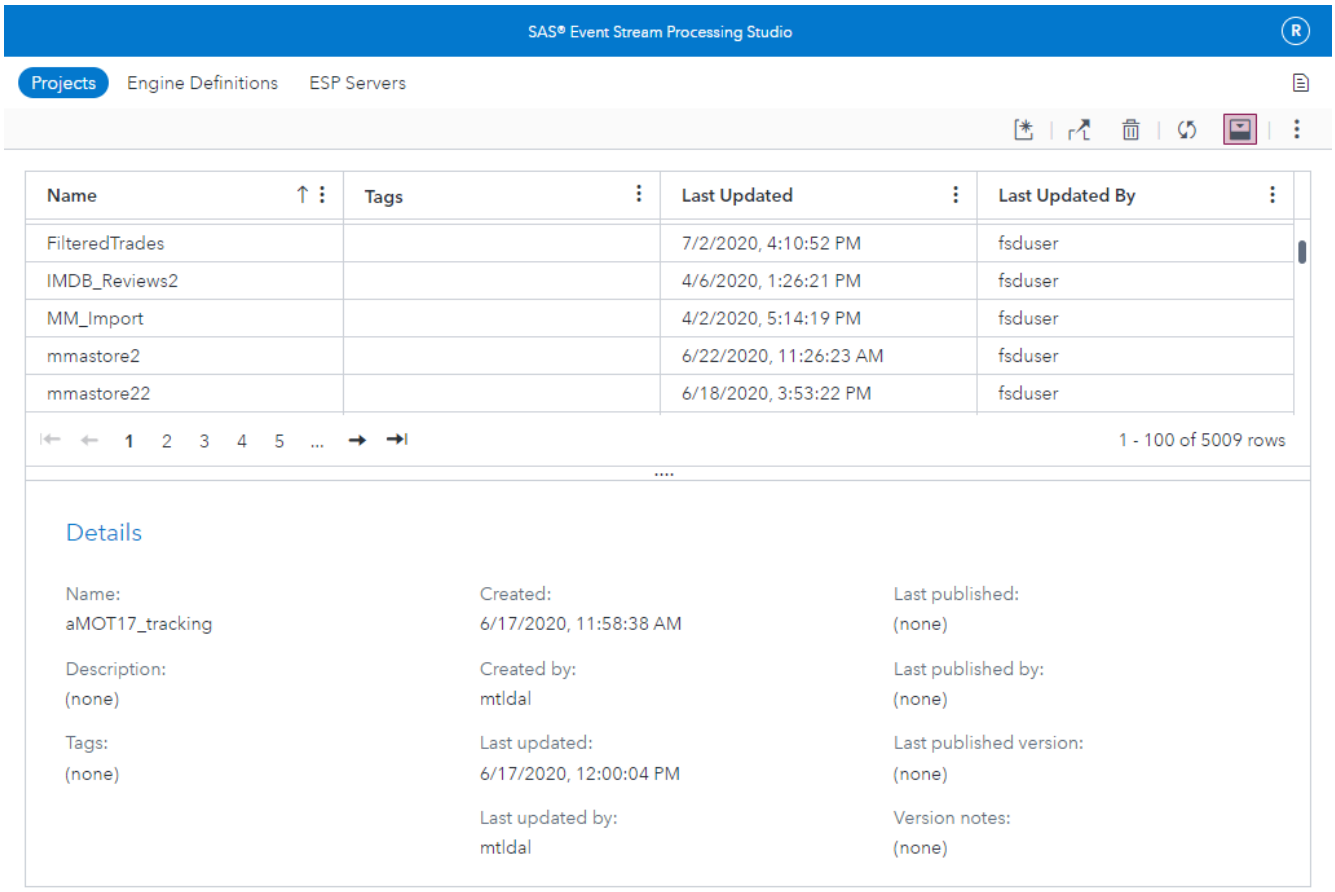
Working with Projects

Overview

A *project* consists of one or more continuous queries. You can use SAS Event Stream Processing Studio to create, upload, download, and delete projects. You can associate your project with a defined engine. For more information, see “Engine Definition Overview” on page 14. The **Projects** page enables you to view the projects in your deployment, along with their identification details and associated engines.

The following figure shows an example:

Figure 6 *The Projects Page*



Note: Projects that are being edited in SAS Event Stream Processing Studio are automatically locked. This ensures that changes to a project cannot be unintentionally overwritten by another user. For more information, see "Project Locking" on page 17.

The **Projects** page displays the following information for each ESP server:

- The project's name.
- Identifying tags assigned to the project.
- The date and time that the project was last updated.
- The user name of the user who last updated a project.

To refresh the main table, click .

To open a project for editing, select the project from the main table and click . Alternatively, you can open a project for editing by double-clicking the project on the main table.

To view a project's additional information, select the relevant project on the **Projects** page.

A tile appears that contains this information, as shown in the previous figure.

To hide this information, click .

Create a 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 unique name for the project.

Note: You must enter a unique project name. Duplicate project names are not supported.

If a project has been published and then subsequently deleted, you cannot reuse the deleted project's name.

b If required, in the **Description** field, enter a description for the project.

c If required, in the **Tags** field, enter any identifying keywords that describe the project.

d Click **OK**.

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

e Click **Yes** to configure an ESP server now or click **No** to configure an ESP server later.

If you selected **Yes**, the Edit ESP Server Properties window appears.

3 If you selected **No** to configure an ESP server later, skip this step. If you selected **Yes** to configure an ESP server now, do the following:

a In the **Name** field, enter or update the name that identifies the ESP server.

b In the **Host** field, enter or update the ESP server's host name or IP address.

c In the **HTTP port** field, enter or update the ESP server's administration port number.

d If required, in the **Description** field, enter or update the description of the ESP server.

e If required, in the **Tags** field, enter or update any keywords that describe the ESP server and then press Enter.

f If required, click **Edit** to change the setting for the **Authentication** field:

The Authentication window appears.

- **None**: This is the default option.
- **Kerberos**: This option is relevant only if the ESP server is configured to require authentication using Kerberos.
- **OAuth token**: This option is relevant only if the ESP server is configured to require authentication. If you select this option, an additional field appears where you must enter the OAuth token.
- **Username and password**: This option is relevant only if the ESP server is configured to require authentication using a user name and password (SASLogon Services). If you select

this option, additional fields appear where you must enter or confirm the user name and password.

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

h If required, select the **Enable server logging** check box to enable logging on the ESP server.

i If required, in the **Number of messages to retain** field, change the default number of messages that are retained by the ESP server log. The default is 10,000 messages.

j Click **OK**.

4 Click **OK**.

SAS Event Stream Processing Studio Modeler appears.

Your project is created with a set of default properties. Before you start creating your model, 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 and select **Save project as**. Enter the relevant information into the Save As window and click **OK**.

Upload a Project

Note: Project XML files uploaded to SAS Event Stream Processing Studio must be encoded in UTF-8 format. Uploading project XML files that are not encoded in UTF-8 format can cause invalid characters to be displayed in SAS Event Stream Processing Studio.

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

The Upload Projects window appears.

2 Click .

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

Note: If you want to upload multiple projects that are located in the same folder, you can select the relevant projects to upload simultaneously. To do this, press and hold Ctrl, select the projects that you want to upload, and click **Open**. If your projects are located in different folders, click again, select the relevant project, and click **Open**.

4 Click **Upload**.

An icon appears. This indicates whether the project was successfully uploaded. Successfully uploaded projects are indicated by the icon . Projects that failed to upload are indicated by the icon .

5 Click **OK**.

The projects that you uploaded appear on the **Projects** page.

Note: You cannot upload a project that has the same name as a project version that has previously been published. This also applies to a project that has the same name as a project version that has been subsequently deleted from SAS Event Stream Processing Studio.

Delete a Project

Select the project that you want to delete from the table on the **Projects** page and click . Click **Yes** to confirm the deletion.

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

Note: Only the working version of a project is deleted. Published versions can still be accessed by other applications through SAS Files and Folders Services.

To delete multiple projects simultaneously, press and hold Ctrl, select the projects that you want to delete, and click . Click **Yes** to confirm the deletion.

Download a Project

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

The project downloads to your computer.

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

Project Metadata

When you create a project, the following unique information that identifies a project is created automatically:

- The user ID of the user who created the project

- The user ID of the user who last modified the project
- The date on which the project was created
- The date on which the project was last modified

The project information is displayed within the `<metadata>` element in your project's XML code, but it is stored in the SAS Event Stream Processing Studio database.

Metadata is also created if you perform one of the following actions:

- Apply a tag to the project
- Make changes to your model in the workspace

  Note: When you make a change to your model in the workspace, a `<meta id="layout">` element is added. This element specifies the name of your model's continuous query, the names of the windows in your model, and each window's X and Y coordinates in the workspace.

- Import a model that contains a SAS Micro Analytic Service module that has been created in SAS Model Manager into SAS Event Stream Processing Studio. For more information, see "Overview" on page 162.

When you publish a version of a project, the following unique information that identifies the version is created automatically:

Note: The following elements are not displayed in the XML code of a working model. However, these elements are displayed in the model's XML code if you have published the model and you are viewing the model in Read-Only mode.

1. The project version's unique ID
2. The project version's major version number
3. The project version's minor version number

Here is an example of a published project version's metadata:

```
<metadata>
    <meta id="studioUploaded">2/26/2018</meta>
    <meta id="studioUploadedBy">espuser</meta>
    <meta id="studioModified">2/26/2018</meta>
    <meta id="studioModifiedBy">espuser</meta>
    <meta id="studioTags">Tag1</meta>
    <meta id="layout">{"cq1":{"Source1":{"x":-310,"y":-315}}}</meta>
    <meta id="studioVersionMajor">1>/meta>
    <meta id="studioVersionMinor">0</meta>
</metadata>
```

In this example, the project's version number is 1.0.

Working with Engines

Engine Definition Overview

An *engine* is the top-level container in the 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 engine definitions. You can associate each project that you produce or upload with an engine definition in SAS Event Stream Processing Studio. The **Engine Definitions** page enables you to view all operational engines in your deployment.

Note: Engine definitions that are being edited in SAS Event Stream Processing Studio are automatically locked. This ensures that changes to an engine definition cannot be unintentionally overwritten by another user. For more information, see "Overview to Locking" on page 16.

The **Engine Definitions** page displays the following information about each engine definition:

- The engine definition's name.
- Identifying tags assigned to the engine definition.
- The date and time at which the engine definition was last updated.
- The user name of the user who last updated the engine definition.

To refresh the main table, click  .

To view additional information for an engine definition, select the relevant engine definition in the main table.

A page appears that contains this information. To hide this information, click  .

To open an engine definition for editing, select the engine definition from main table and click  .

A page appears that contains this information.

Double-clicking an engine definition displays an engine definition page. This page contains:

- A **Name and Description** section – enables you to change the engine definition's name, description, and tags.
- A **Settings** section – enables you to define the engine definition's engine port, HTTP port, and fatal error handling settings. In addition, you can enable execution limits for SAS Micro Analytic Service modules that are written in Python code.
- A **Projects** section – enables 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.

Create a New Engine Definition

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

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:

- 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

5 Associate projects with your engine definition:

a In the **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 +›.

Note: To associate all available projects with your engine definition, click +».

The projects that you have selected appear in the Selected projects table.

c Click **Save**.

The newly associated projects appear in the **Projects** tile.

6 If you changed any of fields on the **Engine Definition** page, click .

Upload an Engine Definition

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

The Upload Engine Definition window appears.

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

3 Navigate to the file that contains 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**.

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 engine definition that you downloaded might vary depending on your browser's configuration.

Delete an Engine Definition

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 **Yes** to confirm the deletion.

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

Note: To delete multiple engine definitions simultaneously, press and hold Ctrl, select the engine definitions that you want to delete, and click . Click **Yes** to confirm the deletion.

Project and Engine Definition Locking

Overview to Locking

Projects and engine definitions that are being edited in SAS Event Stream Processing Studio are automatically locked. This ensures that changes to a project or to an engine definition cannot be unintentionally overwritten by another user. If you open a project or an engine definition, it is assumed that you intend to edit it. If a project or an engine definition has not been locked by another user, the project or the engine definition is locked immediately after you open it. If you are editing a project or an engine definition, the lock is released automatically if you close the tab that contains the project in the application. A lock is also released approximately two minutes after you close the application. For example, if you turn off your computer or close the browser, other users must wait two minutes until they can lock a project or lock an engine definition.

Project Locking

If you attempt to open a project that is locked by another user, you are informed that another user is editing the project. You are prompted to decide whether you want to continue. If you click **Yes**, the project opens in SAS Event Stream Processing Studio Modeler in Read-Only mode. If you click **No**, you return to the **Projects** page.

Read-Only mode enables you to view the model and, if necessary, rearrange the position of the model's windows. However, you cannot save your changes.

Note: Projects are locked against your user name. If you are editing a project, it is not recommended that you open the project in another browser tab or window.

A banner appears when you open a read-only copy of a project. The banner specifies that the user that is editing the project and that a read-only copy of the project has been opened.

The **Windows** pane and magnification icons are unavailable in Read-Only mode. Therefore, you cannot add windows to your model using the **Windows** pane or adjust your model's magnification.

Engine Definition Locking

If you attempt to open an engine definition that is locked by another user, you are informed that another user is editing the engine definition. You are prompted to decide whether you want to continue. If you click **Yes**, the engine definition opens in Read-Only mode. If you click **No**, you return to the **Engine Definitions** page.

Read-Only mode enables you to view the engine definition's properties and to download the engine definition to your computer. However, you cannot modify the engine definition's properties or save your changes.

A banner appears when you open a read-only copy of an engine definition specifying the user that is editing the engine definition. It also specifies that a read-only copy of the engine definition has been opened.

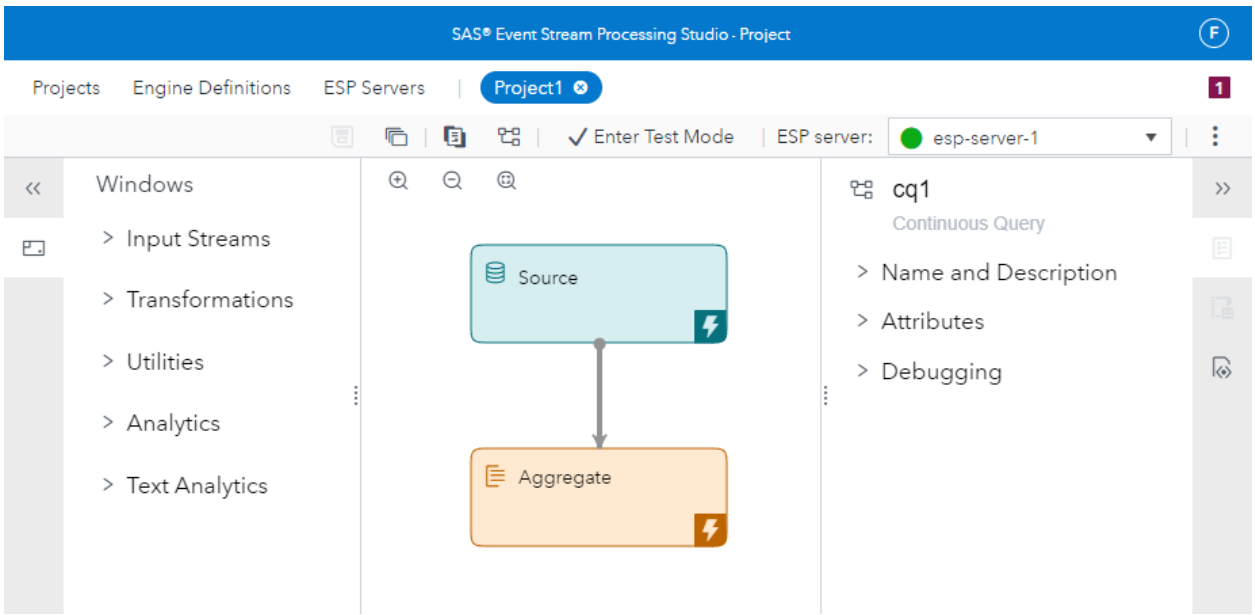
Using SAS Event Stream Processing Studio Modeler

Using SAS Event Stream Processing Studio Modeler

SAS Event Stream Processing Studio Modeler enables you to construct, change, and test SAS Event Stream Processing models. A *model* specifies how an 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.

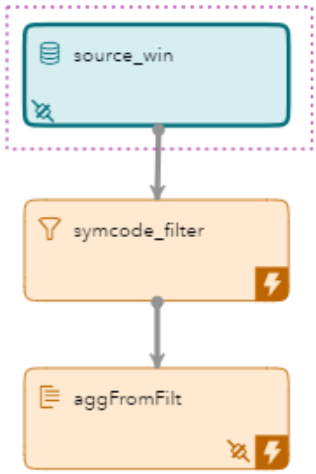
Figure 7 *SAS Event Stream Processing Studio Modeler*



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 .

To select a window, click the window in the workspace. The window that you have selected is shown in the workspace with a dashed bounding box:

Figure 8 *An Example of a Selected Window in the Workspace*



To select multiple windows, click in the workspace, and drag the dashed bounding box around the windows that you want to select.

To pan your model, press and hold Ctrl, click anywhere in the workspace, and then drag the cursor in the appropriate direction.

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 32.

Configure a Model’s ESP Server

If you have not configured any ESP servers, you are prompted to decide whether you want to configure an ESP server. If you decide to create an ESP server, the ESP server that you create becomes the model’s default ESP server and appears in SAS Event Stream Processing Studio Modeler.

If you want to test your model in test mode, you must create an ESP server and assign it to your model. However, creating an ESP server is optional when viewing or editing a model. For more information about testing your model, see “Running a Test” on page 33.

Alternatively, if SAS Event Stream Processing Studio is running on Kubernetes, you can load and run projects on a cluster. When a project is run on a cluster, an ESP server is created on demand. When the project is stopped, the ESP server is deleted from the cluster. Otherwise, an ESP server in a cluster behaves in the same way as an ESP server that is not in a cluster. For more information, see “Working with a Kubernetes Cluster in SAS Event Stream Processing Studio” on page 40.

Functionality is limited if you open a model that does not have an assigned ESP server. For example, connector properties are unavailable to models for which ESP servers are not assigned. You can manage your ESP servers on the **ESP Servers** page. If you have configured multiple ESP servers, you can use this page to change the default ESP server that is associated with your model.

When an ESP server is associated with your project, the association is saved in your browser's local storage. The default ESP server selected on the **ESP Servers** page applies only to projects that do not have an associated ESP server within your browser's local storage. For example, if you have not opened the project before, the default ESP server is associated with the project.

The **ESP Server** drop-down list displays the ESP server that is associated with your project. ESP servers in a Kubernetes cluster are identified in the drop-down list with the following icon: icon. For example, the following ESP server is in a Kubernetes cluster:

For more information about managing ESP servers, see “Managing ESP Servers in SAS Event Stream Processing Studio” on page 37.

Add a Window

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

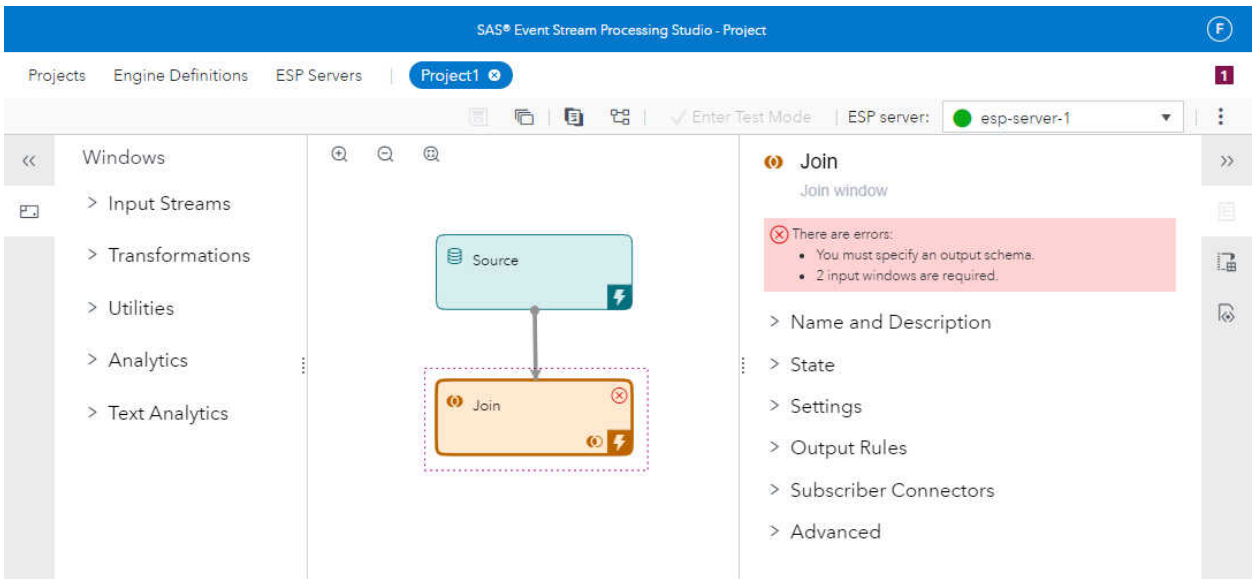
Note: Alternatively, you can add a window to the continuous query by double-clicking the relevant window in the **Windows** pane.

The windows are grouped into the following categories:

- Input Streams
- Transformations
- Utilities
- Analytics
- Text Analytics

You must ensure that you enter valid properties for each window. Entering invalid window properties causes the window to display an error icon: . Here is an example of a model where the Join window contains invalid properties:

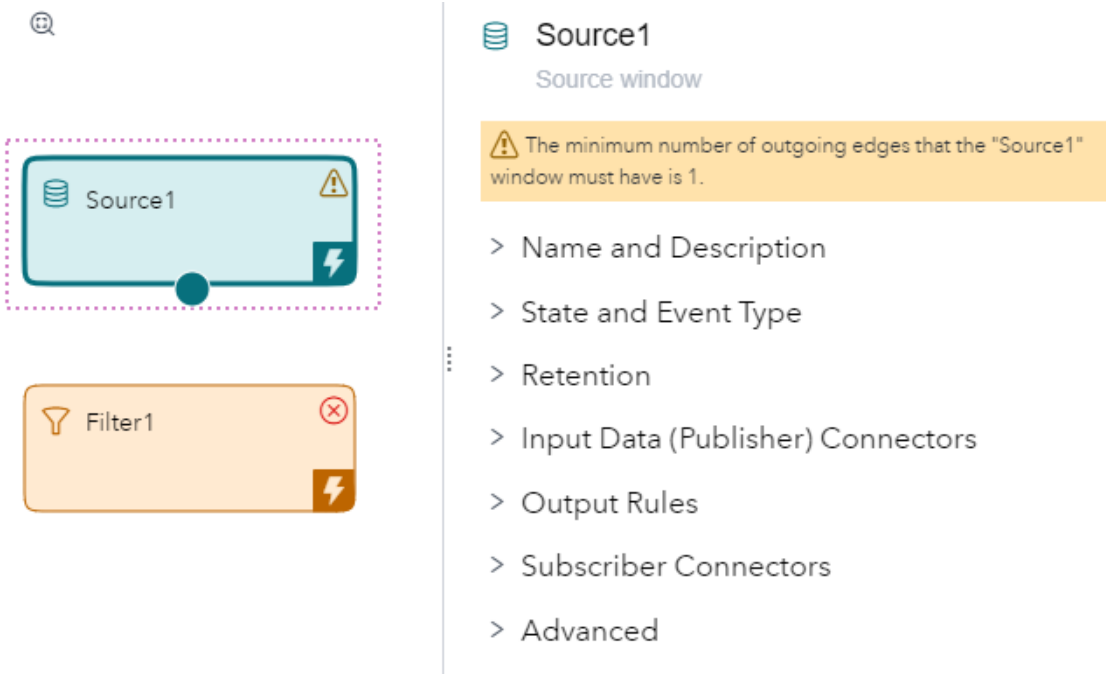
Figure 9 *A Window Validation Error Icon and Corresponding Message*



In addition, if a window requires further changes for the model to run successfully, the window displays a warning icon: ⚠. The right pane that displays the window's properties shows the corresponding warning message.

Here is an example of a model where the Source window requires further changes for the window to be in a valid state:

Figure 10 *A Window Validation Warning Icon and Corresponding Message*



Connecting Windows

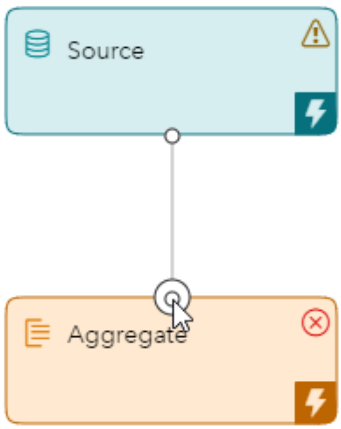
To connect a window to another window with an edge:

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

Figure 11 *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:

Figure 12 *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**.

Edge Display Types

Connecting edges can appear differently depending on the type of connection between windows.

For information about edge display types, see the following table:

Edge Display Type	Connecting Edge	Details
Event stream		Event stream edges connect input windows that contain event stream data.
Supporting data		Supporting data edges connect input windows that contain geometric data (that is, where the edge role is set to `Geometry`). They can also connect secondary input windows to a Join window.
Non-data edges		Non-data edges connect windows that do not contain event stream data (that is, where the edge role is set to `Model` or `Request`).

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

Edge Roles

You can use SAS Event Stream Processing Studio Modeler to configure the edge roles of connecting edges in your model. Edge roles must be specified for edges that connect streaming analytics windows and for edges that connect Geofence and Join windows. Each edge is assigned a role by default.

To change an edge's default role:

1. In the workspace, select the edge whose role you want to change.
2. In the right pane, in the **Role** field, change the default selection.

For information about edge roles, see the following table:

Window Name	Default Edge Role	Available Edge Roles	Dependencies
Join	Left table	Left table or Right table	If you assign an edge role to a Join window's connecting edge, the remaining connecting edge is automatically assigned the alternative edge role.
Geofence	Position	Position or Geometry	If you assign an edge role to a Geofence window's connecting edge, the remaining connecting edge is automatically assigned the alternative edge role.
Model Reader	Request	Request	Not applicable

Window Name	Default Edge Role	Available Edge Roles	Dependencies
Model Supervisor	Request	Request or Model	Not applicable
Calculate	Data	Data or Request	Not applicable
Train	Data	Data or Request	Not applicable
Score	Data	Data or Model	If you assign an edge role to a connecting edge, the remaining connecting edge is automatically assigned the alternative edge role.

Delete a Window or an Edge

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:

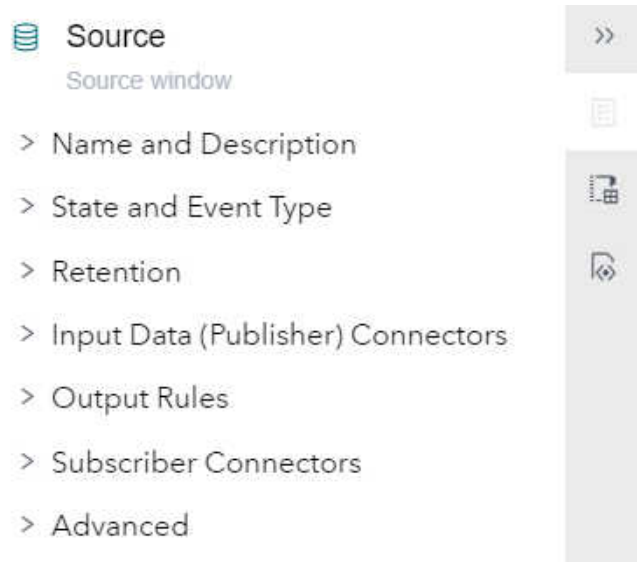
Icon	Description
	Indicates that the window contains an error message that will cause the model to fail to run.
	Indicates that the window requires further changes for it to be in a valid state.
	Indicates that the window contains a fully stateful index that is stored in memory. **Note:** If the window contains a non-stateful index, this icon is not displayed. This color of this icon changes depending on the window that contains it. This example shows an icon on a Source window.

Icon	Description
	Indicates that the window contains a fully stateful index that is stored on disk. **Note:** If the window contains a non-stateful index, this icon is not displayed. This color of this icon changes depending on the window that contains it. This example shows an icon on a Source window.
	Indicates that the Source window contains either a publisher connector or a subscriber 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.

Configure the Properties of a Window

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

Figure 13 *Properties of the Source Window*



Note: If the right pane displays XML code or an output schema, you can view the window's properties by clicking  in the right pane to display the properties instead.

Validate an Expression

You can validate expressions that you enter in SAS Event Stream Processing Studio Modeler. This functionality validates the syntax of the code that you enter. It can be particularly useful when entering large blocks of codes or pasting code from an external source. DataFlux expressions cannot be validated on an ESP server that is in a cluster. For information about clusters, see "Working with a Kubernetes Cluster in SAS Event Stream Processing Studio" on page 40. To validate expressions, a working ESP server must be available. If you are using only a Kubernetes environment, expression validation is not supported for a project running on a cluster. When expression validation is available in the application, the  button is shown. After you click  , a message appears, informing you of the expression's validity. Expression validation is available for the following operations:

- defining filter conditions in Filter windows
- performing non-key field calculations in Compute and Join windows
- window-output splitter-slot calculations (for example, you can use an expression to evaluate where to send a generated event)

Use the Expression Editor

If you are entering large blocks of code or pasting large blocks of code from an external source, it is recommended that you use the Expression Editor. When the Expression Editor is available in the application, the  icon is shown next to the relevant field.

The Expression Editor contains two panes:

- The left pane contains two tabs: **Functions** and **Schema**.
  - The **Functions** tab enables you to search for and add Expression Engine Language (EEL) functions to your expression. To do this, click and hold the function and drag it to the relevant location in your code.
  - The **Schema** tab enables you to add schema fields to your expression. To do this, click and hold the schema field and drag it to the relevant location in your code.

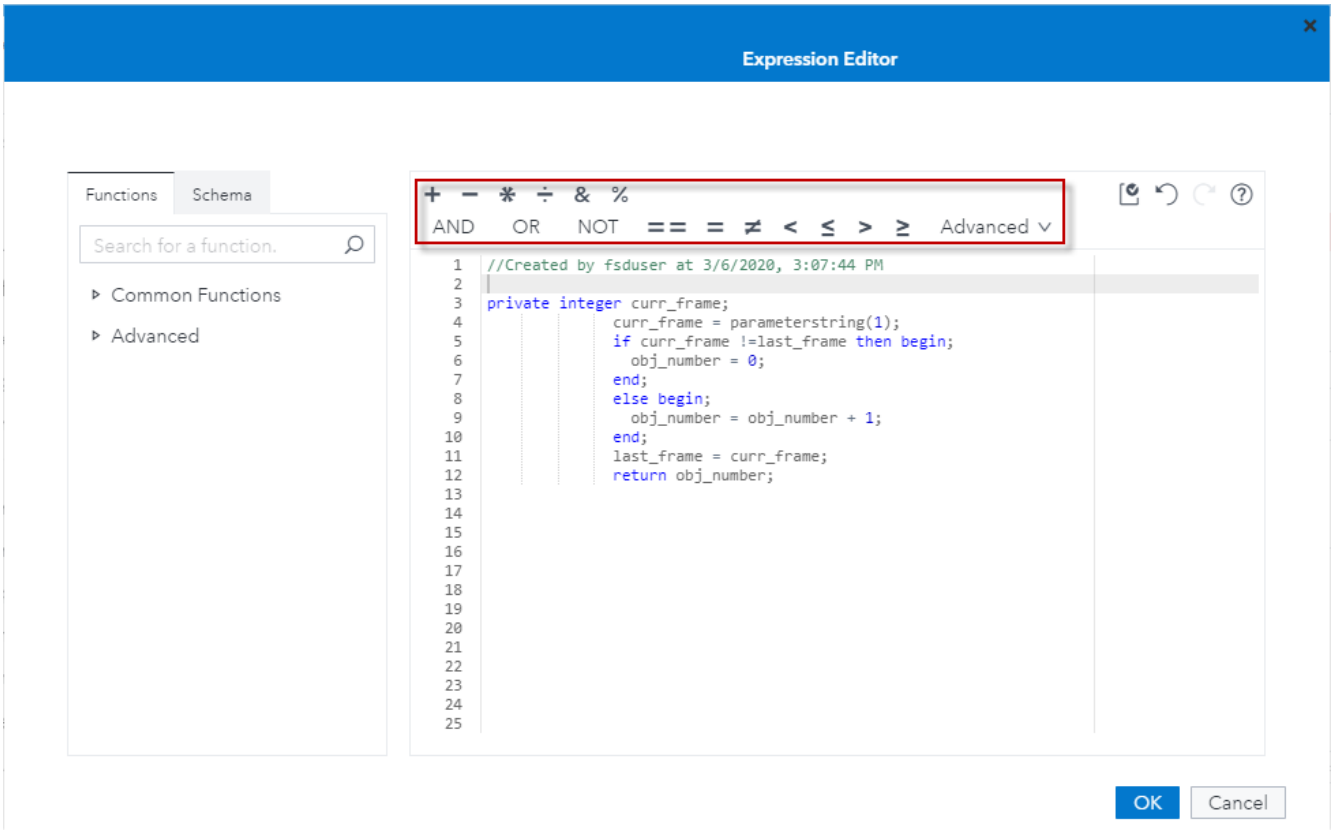
  To search for a function or a schema field, enter your search term in **Search** field in the corresponding tab.
- The right pane contains an editor that enables you to manually enter Expression Engine Language (EEL). For more information, see "Overview of the Expression Engine Language" in *Expression Language: Reference Guide*. This functionality validates the syntax of the code that you enter and highlights the syntax using a color scheme.

The Expression Editor contains icons that enable you to perform specific operations on your code. For information about these icons, see the following table:

Icon	Description
	Reverts your previous change.
	Reverts the effects of the undo action.
	Validates your expression in the Expression Editor. After the validation has completed, a message appears, informing you of the expression's validity.
	Indicates that your expression contains an error. The Expression Editor displays a message specifying the error and its location. For a more detailed description of the error, position the cursor over the icon.

The Expression Editor enables you to include operators in combination with numbers, strings, functions, and functions that use other functions. You can access these operators from the toolbar. For more information, see "Operators" in *Expression Language: Reference Guide*. For a description of the operator, position the cursor over the relevant operator on the toolbar. The toolbar is highlighted with a red box in the example below:

Figure 14 *An Example of an Expression in the Expression Editor*



The editor enables you to select and insert existing code elements to complete code that you have partially entered. After partially entering code in the editor, you are presented with a list of

suggestions for automatic completion. To accept the first suggestion from the list, press Tab. To accept one of the alternative suggestions, press the down arrow key until the relevant suggestion is selected and then press Tab.

Using the XML Editor

Using the XML Editor

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.

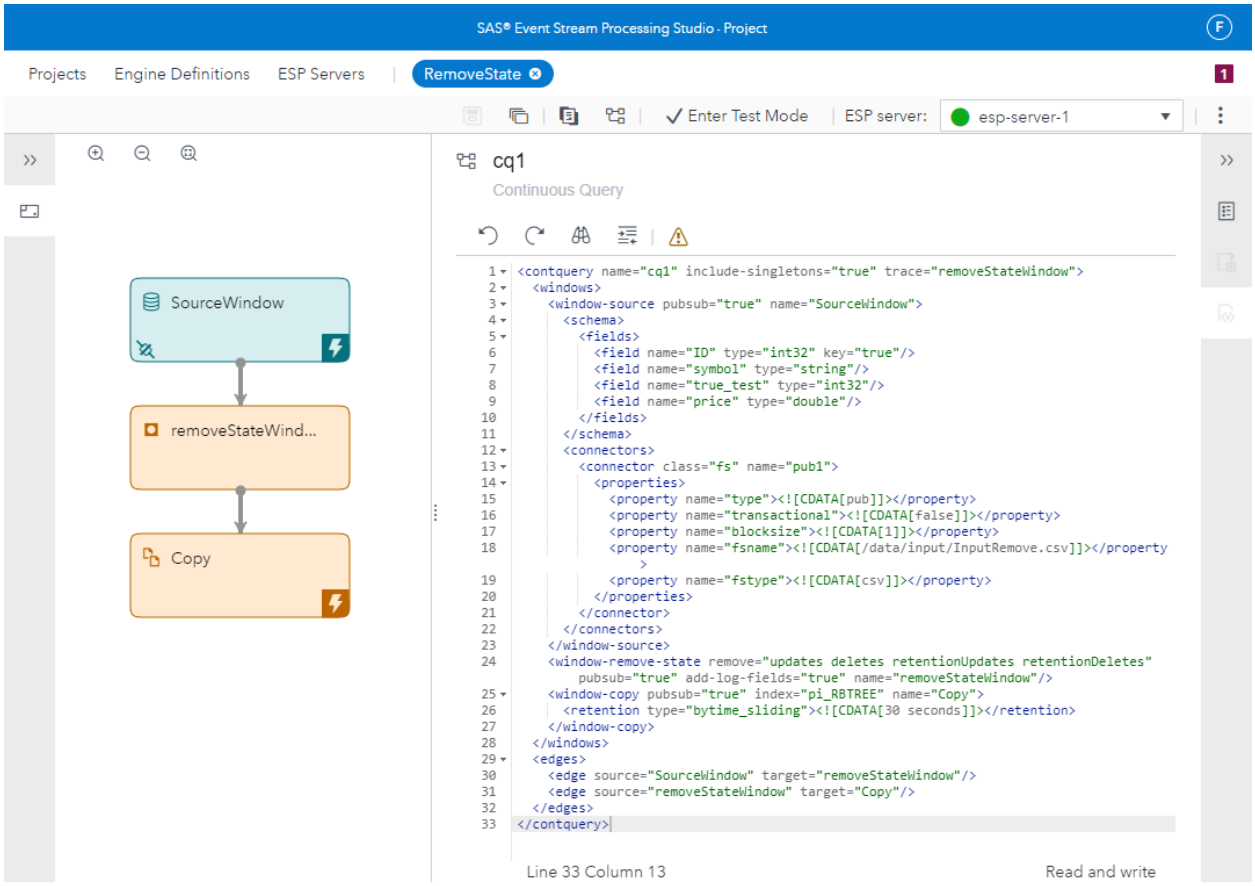
CAUTION

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  on the right toolbar.

The right pane displays the XML Editor.

Figure 15 *The XML Editor*



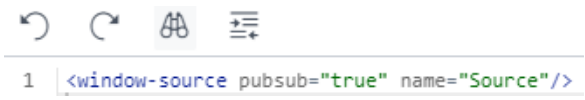
Selecting a specific element in your workspace reloads the XML Editor to display only the corresponding section of XML code. To display the entire project's XML again, click . If you have associated a project with an engine, the XML code that specifies the engine is not included in the project's XML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

If you selected a specific ESP window, 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 add a comment to your XML code that is not enclosed within its relevant XML element, the comment is automatically moved within the XML element. This occurs when the XML code is reordered. For example, the XML code is reordered if you save your model or if you move away from the XML editor.

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:

Figure 16 *The XML Editor Displaying a Source Window's XML Code without the Window's Default State*



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

Figure 17 *The XML Editor Displaying a Source Window's XML Code with the Window's Default State*

Unique identifying project information is displayed within the `<metadata>` element in your project's XML code. Although the metadata is displayed in your project's XML code, it is located in the SAS Event Stream Processing Studio database. For more information, see "Project Metadata" on page 12.

Using Editing Tools and Keyboard Shortcuts

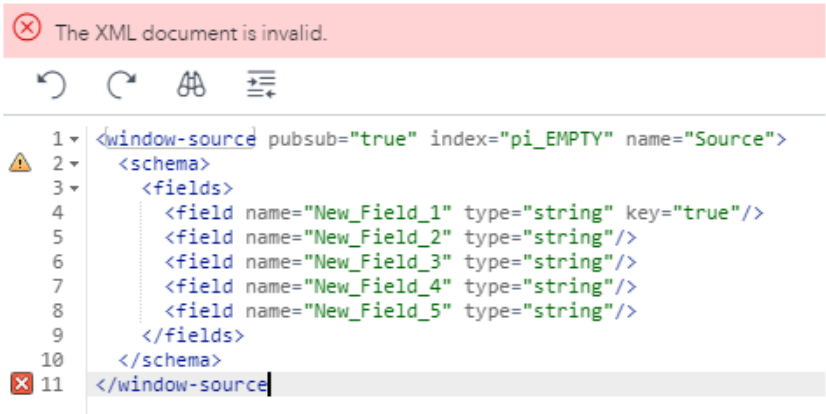
The XML Editor includes a toolbar that contains editing tools. These tools are also accessible using keyboard shortcuts.

Icon	Action	Keyboard Shortcut
	Reverts your previous change	Ctrl + Z
	Reverts the effects of the undo action	Ctrl + Y
	Prompts you to search for specific text. Pressing Ctrl + F again prompts you to replace the text that you have searched for.	Ctrl + F
	Formats the XML code that you have manually entered	Not available
No icon	Removes the code that you have selected from its original position	Ctrl + X
No icon	Copies the code that you have selected to the clipboard	Ctrl + C
No icon	Pastes the code on the clipboard at the cursor's position	Ctrl + V
No icon	Selects all of the code in the XML editor.	Ctrl + A

Validation

The XML Editor automatically validates the syntax of the code that you enter. If you enter invalid code, the XML Editor displays the following error message:

Figure 18 *Invalid XML Warning*



The XML error message also indicates the error's location with a breadcrumb trail. Each section of the breadcrumb trail represents an XML tag in your XML code. The top-level XML tag in your XML code is not included in the breadcrumb trail.

Position the cursor over the warning icon ⚠ to view a generic description of the error in your XML code. The icon ⊗ also displays the location of the error in your XML code.

For a more detailed description of the error, position the cursor over the icon ⊗.

Efficiency Tips

For some window types, you can copy schema fields between windows 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 contain schemas that are implied or have been internally generated.

Continuous queries

Configuring Continuous Queries in SAS Event Stream Processing Studio

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

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.

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.

Testing Models in SAS Event Stream Processing Studio

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 event streams that can be consumed by subscribers.

Note: An *engine* is the top–level container in the 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.

For information about how to run a test, see “Running a Test” on page 33.

You can also analyze your model’s performance in real-time by viewing its log. For more information, see “Viewing a Model’s Test Logs in SAS Event Stream Processing Studio” on page 36.

Running a Test

1 On the **Projects** page, right-click the project that contains the model that you want to test.

2 Select **Open project** from the menu.

The project appears in a new page.

Note: Ensure you have saved any changes that you have made to the project. Projects that contain unsaved changes cannot be opened in test mode.

3 Click ✓ Enter Test Mode .

A page appears, enabling you to test your model.

4 In the **ESP server** drop-down list, verify that an ESP server has been selected to perform the test on.

Note: If SAS Event Stream Processing Studio does not contain any registered ESP servers, you must register one before you can continue testing your model. You can register a new ESP server on the **ESP Servers** page. When a project is contained within a kubernetes cluster, an ESP server is created on demand in the cluster. When the project is stopped, the ESP server is deleted from the cluster. For more information, see “Working with a Kubernetes Cluster in SAS Event Stream Processing Studio” on page 40.

5 To configure your test’s settings, click ⋮ and select **Output data settings**.

The Output Data Settings window appears.

6 To return events to test mode in real time, select **Return events from server as they happen**. Alternatively, to return events to test mode in pages, select **Return events from server in pages**.

Note: This version of SAS Event Stream Processing Studio uses the WebSocket protocol to subscribe to windows. Models that are executed might display events in a different sequence than models that were executed in previous versions. 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.

If you selected **Return events from the server in pages**, do the following:

a In the **Maximum page size (events)** field, enter the maximum number of events to be displayed in a results page.

Note: To prevent an excessive number of results being simultaneously returned from the ESP server, you can display results in paged format. This can improve your browser's performance. However, if the total number of results is greater than maximum page size, some results are not displayed.

b In the **Interval (ms)**, enter an interval at which each page is to be returned from the ESP server (in milliseconds).

7 Click **OK**.

8 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. To do this, click ⋮ and select **Tile**.

9 From the list of windows on the left of the screen, select the windows whose results you want to view.

10 To run the test, click ▷ Run Test.

You can group information by column. The results table contains a horizontal bar at the top of the table, with the text **Drag a column header here to group by that column**. To group information by column, drag a column heading to the bar. If required, you can drag additional columns to the bar.

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

The Status indicator informs you of your test's current status. Your test can have the following statuses:

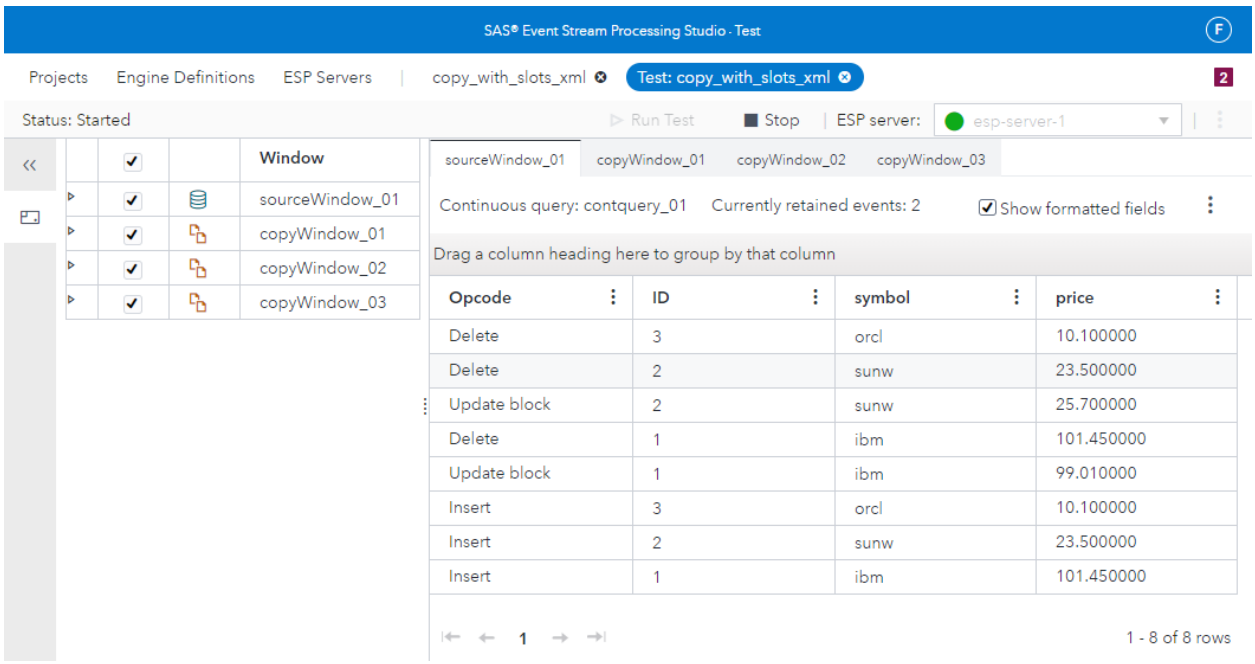
Table 1 *Table of Test Statuses*

Status	Description
Initial	The test is in an initial state and has not started.
Starting	The test is starting.
Started	The test has been started and is running.
Stopping	The test is stopping.
Stopped	The test has been stopped.

You can use the **Show formatted fields** check box to choose whether data appears exactly as it was received from the ESP server or with additional formatting. Here are some examples of additional formatting that is applied when the check box is selected:

- Dates are shown as Coordinated Universal Time (UTC) in ISO 8601 format, for example, 2018-11-30T13:33:47.000Z. If you clear the check box, dates appear in UNIX Epoch time, as this is the format in which the data is received from the ESP server.
- A dot is used as a separator in certain types of numerical data, rather than another separator, such as a comma. If you clear the check box and your locale is set to a locale that uses another separator, that separator is displayed instead of a dot.
- Opcodes are displayed using their localized names if your locale is not set to an English-language locale. If you clear the check box, opcodes are always shown in English, as this is how the data is received from the ESP server.

Figure 19 *Test Mode*



Note: If a window in your model contains more than 1,000 results, only the last 1,000 results are displayed in test mode.

The left pane contains details of your model's windows and their output schema fields. An icon appears next to each window name and output schema field identifying the window type or field type. To identify a window type or a field type, position the cursor over the associated icon.

All output schema fields appear by default for each window in your model. However, you can filter these fields to ensure that only specific fields appear. To do this, in the left pane, click ▷ in the row for the window whose fields you want to filter. Deselect the fields that you do not want to appear. To deselect all fields in the window's output schema, click . If a window in your model contains more than 15 fields in its schema, only the first 15 fields that you have selected are shown in test mode.

11 To stop the test, click ■ Stop.

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

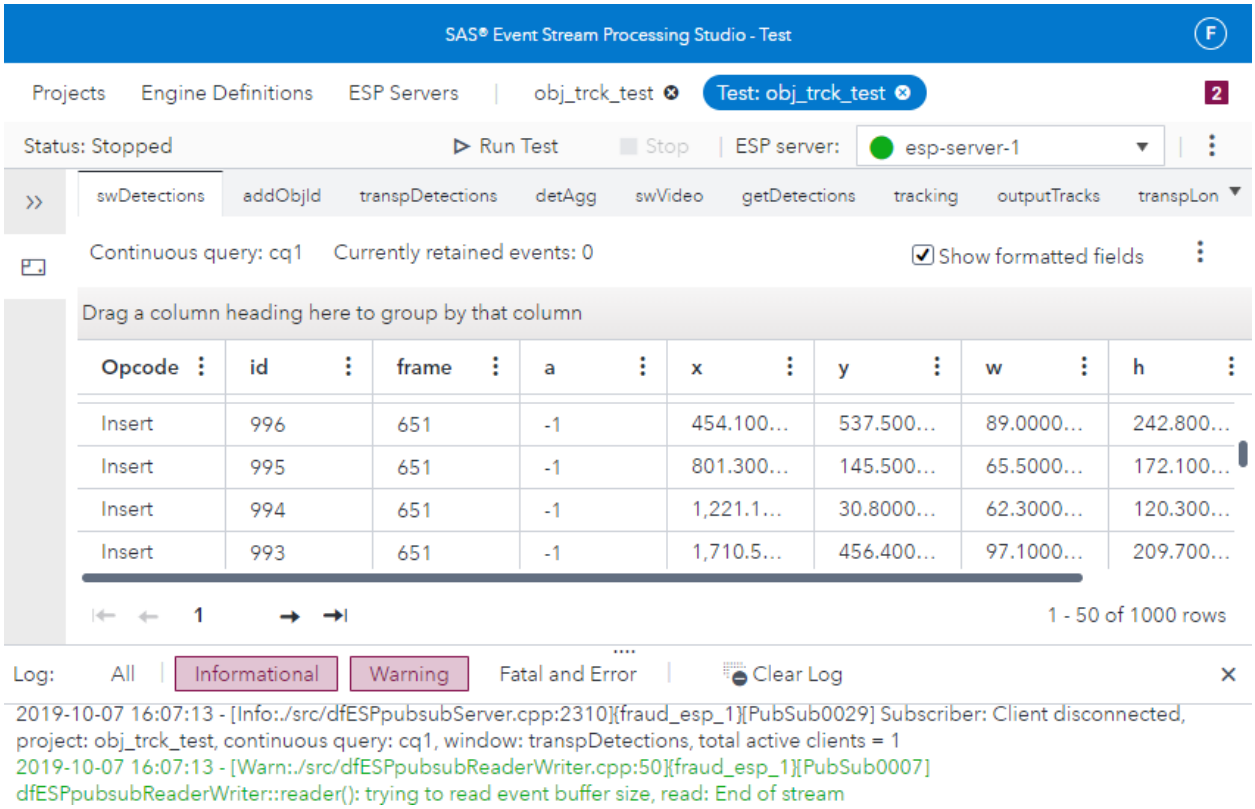
Note: If the test fails and the resulting error message does not explain what caused the failure, you can troubleshoot the problem by checking the test logs in the Log pane. For more information, see "Viewing a Model's Test Logs in SAS Event Stream Processing Studio" on page 36.

Viewing a Model's Test Logs in SAS Event Stream Processing Studio

You can monitor your model in real-time by viewing the model's logs in test mode. Log messages appear in the Log pane, a horizontal pane located at the bottom of test mode. Test mode logging is disabled by default on each ESP server. To enable test mode logging on the ESP server that you are currently using, click **Enable** in the Log pane.

Historical log messages are not displayed. SAS Event Stream Processing Studio displays messages that were logged after you ran the model in test mode. If you select another ESP server or close test mode for the specific model, log messages that were displayed in SAS Event Stream Processing Studio are not retained.

Figure 20 *Test Mode with Logging Enabled*



To filter log messages by message type, select one or more of the following options in the Log pane:

- All – Shows all log message types.
- Informational – Shows general log messages that are not warnings or errors. For example, in Figure 24, only informational and warning messages are shown.

- Warnings – Shows only warning messages.
- Fatal and Error – Shows only error messages, including fatal errors and normal errors.

To clear the contents of the Log pane, click Clear Log.

To close the Log pane, click ×. To reopen the Log pane, click ⋮ and select **Log** from the drop-down list.

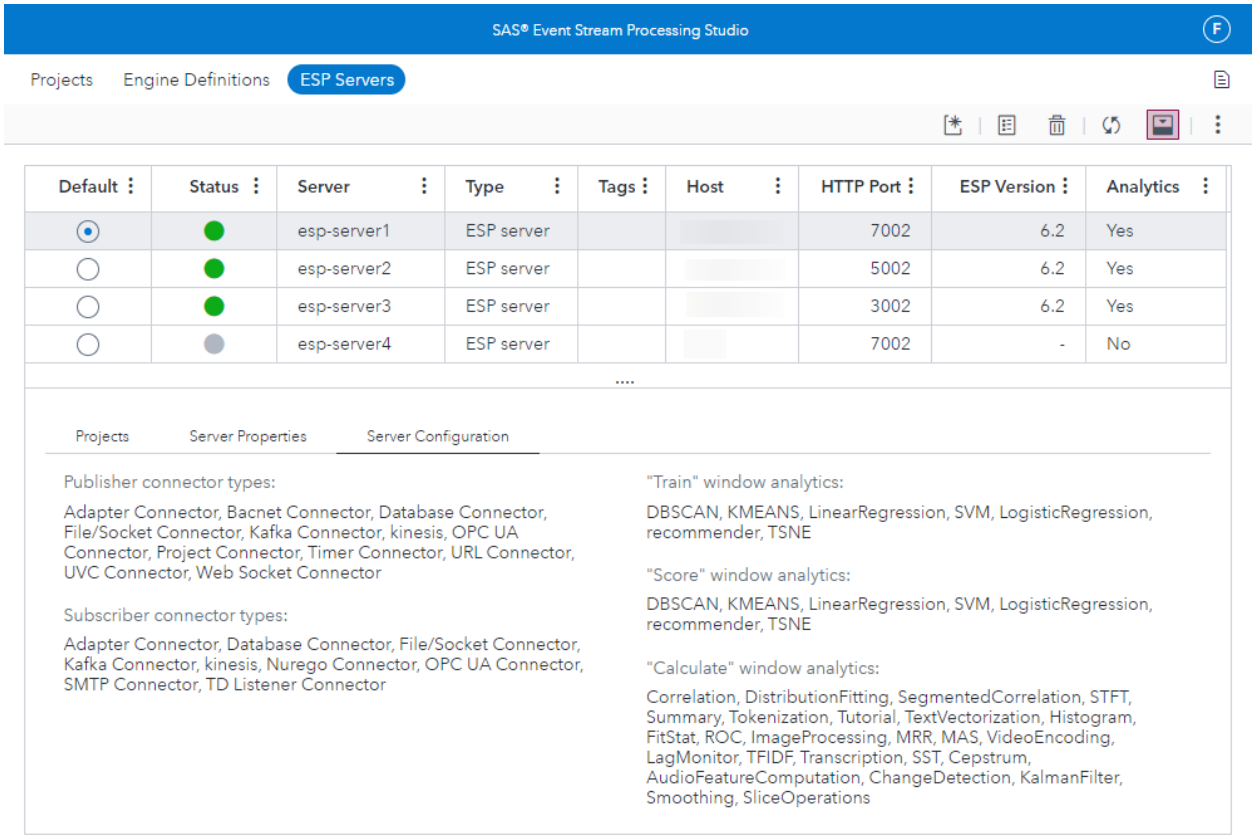
Managing ESP Servers in SAS Event Stream Processing Studio

Managing ESP Servers in SAS Event Stream Processing Studio

You can use the **ESP Servers** page to view details of existing ESP servers in your deployment. You can also use the controls on this page to create, edit, and delete ESP servers.

The following figure shows an example:

Figure 21 *The ESP Servers Page*



The **ESP Servers** page displays the following information for each ESP server:

- Whether it is the default ESP server.
- The ESP server's status.
- The ESP server's name.
- The ESP server type:
  - ESP server: The ESP server is not in a Kubernetes cluster
  - Cluster server: The ESP server is in a Kubernetes cluster
- Identifying tags assigned to the ESP server.
- The host on which the ESP server is running.
- The port that is used for HTTP administration requests.
- The SAS Event Stream Processing version installed on the host on which the ESP server is running.
- Whether streaming analytics is available on the ESP server.

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

Icon	Description
	Undefined – The ESP server status has yet to be established.
	Available – The ESP server is available and operating normally.
	Available in cluster – The ESP server is in a Kubernetes cluster, available, and operating normally.
	Unavailable – The ESP server is not available.

You can group information by column. To enable grouping, click ⋮ and select **Group columns**.

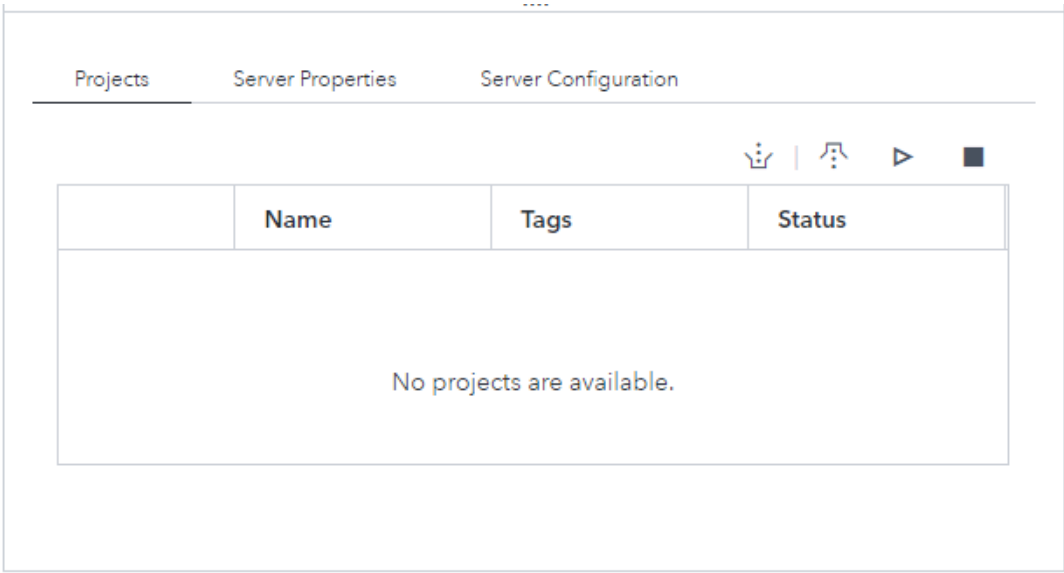
A horizontal bar at the top of the table appears. The bar contains the text **Drag a column header here to group by that column**. To group information by column, drag a column heading to the bar. If required, you can drag additional columns to the bar. To refresh the table of ESP servers, click ↻.

To view additional information for an ESP server, select the relevant ESP server on the **ESP Servers** page.

To hide this additional information, click

.

Figure 22 *The Projects Tab*



The **Projects** tab appears by default and specifies the projects that have been uploaded to the ESP server selected.

You can use the **Projects** tab to load, unload, start, and stop a project on the ESP server.

To load a project onto the ESP server that is not in a cluster, click . In the Load Project window, select the project that you want to load from the table and click **OK**. This process uploads a working copy of the project to the ESP server. For information about loading and starting a project on an ESP

server that is in a cluster, see “Load and Start a Project on an ESP Server That Is in a Cluster” on page 42. For information about stopping a project on an ESP server that is in a cluster, see “Stop a Project from an ESP Server That is in a Cluster” on page 42.

Note: Uploading projects that have already been published is not permitted.

After you have loaded the project onto the ESP server, you can run the project by clicking . The **Running Projects** field on the corresponding row in the table on the **ESP Servers** page is updated with the running project’s name. To stop the running project, click . To unload the project from its ESP server, click .

The **Server Properties** tab in the bottom pane contains information about the ESP server’s properties, such as the authentication method used on the ESP server. For example, an ESP server can use Kerberos, OAuth, or SASLogon services authentication. This tab also displays information such as the ESP server’s host name, HTTP port, and whether Transport Layer Security (TLS) is enabled on the ESP server. For more information, see “Create or Edit an ESP Server Not in a Cluster” on page 41. To edit the ESP server’s properties, click **Edit properties**. The Edit ESP Server Properties window appears, enabling you to do this.

The **Server Configuration** tab contains information about connector types, streaming analytics, and whether SAS Event Stream Processing has been enabled to meter the number of events that are processed on the ESP server.

Working with a Kubernetes Cluster in SAS Event Stream Processing Studio

SAS Event Stream Processing Studio includes the ability to load and run projects on a Kubernetes cluster. When a project is loaded and started within a cluster, an ESP server is created on demand in the cluster. When the project is stopped, the ESP server is deleted from the cluster. Otherwise, an ESP server in a cluster behaves in the same way as an ESP server that is not in a cluster.

From the **ESP Servers** page, you can load and start a project on an ESP server in a cluster. For more information about starting a project on an ESP server in a cluster, see “Load and Start a Project on an ESP Server That Is in a Cluster” on page 42. You can also stop a project on an ESP server in a cluster. For more information about stopping a project on an ESP server in a cluster, see “Stop a Project from an ESP Server That is in a Cluster” on page 42.

You can create ESP servers that are associated with a cluster and ESP servers that are not in a cluster. In an environment where the ability to deploy a project to a cluster is enabled, SAS Event Stream Processing Studio and SAS Event Stream Manager are running as stand-alone applications (deployed without SAS Viya services). This means that you must manually upload projects to SAS Event Stream Processing Studio. For more information, see “Upload a Project” on page 11.

Note: DataFlux expressions cannot be validated on an ESP server that is in a cluster. For more information about expression validation, see “Validate an Expression” on page 26.

Create or Edit an ESP Server Not in a Cluster

This topic applies to ESP servers that are not in a cluster. For information about ESP servers in a cluster, see "Working with a Kubernetes Cluster in SAS Event Stream Processing Studio" on page 40.

1 If you want to create a new ESP server, on the **ESP Servers** page, click  .

The ESP Server Properties window appears.

Or

If you want to edit an existing ESP server's details, on the **ESP Servers** page, select the ESP server that you want to open and click  .

The Edit ESP Server Properties window appears.

2 Do the following to create or edit an ESP server:

a In the **Name** field, enter or update the name that identifies the ESP server.

b In the **Host** field, enter or update the ESP server's host name or IP address.

c In the **HTTP port** field, enter or update the ESP server's administration port number.

d If required, in the **Description** field, enter or update the description of the ESP server.

e If required, in the **Tags** field, enter or update any keywords that describe the ESP server and then press Enter.

f If required, click **Edit** to change the setting for the **Authentication** field:

The Authentication window appears.

- **None**: This is the default option.
- **Kerberos**: This option is relevant only if the ESP server is configured to require authentication using Kerberos.
- **OAuth token**: This option is relevant only if the ESP server is configured to require authentication. If you select this option, an additional field appears where you must enter the OAuth token.
- **Username and password**: This option is relevant only if the ESP server is configured to require authentication using a user name and password (SASLogon Services). If you select this option, additional fields appear where you must enter or confirm the user name and password.

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

h If required, select the **Enable server logging** check box to enable logging on the ESP server.

i If required, in the **Number of messages to retain** field, change the default number of messages that are retained by the ESP server log. The default is 10,000 messages.

j Click **OK**.

Load and Start a Project on an ESP Server That Is in a Cluster

For an introduction to using a Kubernetes cluster with SAS Event Stream Processing Studio, see "Working with a Kubernetes Cluster in SAS Event Stream Processing Studio" on page 40.

1 On the **ESP Servers** page, click  and select **Load and start project in cluster**.

   The Load and Start Project in Cluster window appears.

2 In the **Project** field, select the project that you want to load onto the ESP server from the drop-down list.

3 The **Use default settings** check box is selected by default. To use the default settings, leave the check box selected. To define your own custom settings, clear the **Use default settings** check box and click **Edit deployment settings**. Before attempting to define your own custom settings, contact your system administrator for advice.

   If you clicked **Edit deployment settings**, the Deployment Settings window appears.

   a In the **Requests** section, specify the amount of memory and CPU that you want to allocate to the ESP server when it is created.

   b In the **Limits** section, specify the maximum amount of memory and CPU that the ESP server is allowed to use.

   c In the **Autoscale** section, specify the minimum and maximum number of ESP server instances (replicas) that you want to allow.

   d Click **OK**.

4 Click **OK**.

   A new ESP server is created. The project that you loaded is started on the ESP server.

Stop a Project from an ESP Server That is in a Cluster

When you stop a project that is running on an ESP server that is in a cluster, the ESP server is deleted from the cluster. For an introduction to using a Kubernetes cluster with SAS Event Stream Processing Studio, see "Working with a Kubernetes Cluster in SAS Event Stream Processing Studio" on page 40. For information on how to stop projects on ESP servers that are not in a cluster, see "Managing ESP Servers in SAS Event Stream Processing Studio" on page 37.

To stop a project that is running on an ESP server that is in a cluster:

1 On the **ESP Servers** page, select the ESP server that contains the project that you want to stop.

2 In the **Project** section in the bottom pane, select the project that you want to stop, and click  .

   The project is stopped and the ESP server is deleted from the cluster.

Delete an ESP Server That Is Not in a Cluster

Deleting an ESP server removes it from the table on the **ESP Servers** page. Deleting ESP servers can be useful if the table contains ESP servers that are no longer used. You can delete an ESP server that is still running. To delete a specific ESP server from the table:

1 On the **ESP Servers** page, select the ESP server that you want to delete and click  .

   The Remove ESP Server window appears.

2 Click **Yes** to confirm the deletion of the ESP server.

Note: To delete multiple ESP servers simultaneously, press and hold Ctrl, select the ESP servers that you want to delete, and click  . Click **Yes** to confirm the deletion.

Delete an ESP Server That Is in a Cluster

To delete an ESP server that is in a Kubernetes cluster, stop the project that is running on the ESP server. Stopping the project deletes the ESP server from the cluster permanently. Alternatively, select the ESP server that you want to delete from the table on the **ESP Servers** page and click  . Click **Yes** to confirm the deletion of the ESP server. For more information about how to stop an ESP server that is in a cluster, see “Stop a Project from an ESP Server That is in a Cluster” on page 42.

Publishing Project Versions

Publishing Project Versions

You can use SAS Event Stream Processing Studioto create and manage multiple versions of a project. If your deployment contains a running instance of SAS Event Stream Manager, publishing a project version from SAS Event Stream Processing Studio makes the project version available to SAS Event Stream Manager.

Note: If SAS Event Stream Processing Studio is running as a stand-alone application (deployed without Viya services), you cannot publish projects. In this scenario, if SAS Event Stream Manager is installed in your deployment, a project cannot be published to SAS Event Stream Manager and is therefore not visible. To make a project visible in SAS Event Stream Manager, use SAS Event Stream

Processing Studio to download the project. For more information, see “Download a Project” on page 12. You must then manually upload the project to SAS Event Stream Manager.

You cannot edit project versions after they have been published. A project that you can edit is designated as the working copy of the project. It does not become a project version until you publish it. The working copy of the project enables you to make changes to the project without affecting any project versions that you previously published. You must then manually upload the project to SAS Event Stream Manager.

When you publish a project version, the version’s XML code is updated to display its unique ID number. Project versions that are published for the first time are assigned a version number of 1.0. The number to the left of the decimal point is the project’s major version number. The number to the right of the decimal point is the project’s minor version number. Publishing subsequent versions of a project increments the major version number. In SAS Event Stream Processing Studio, you can publish major project versions, but you cannot publish minor project versions. Minor versions are created when you make a change to the project version in SAS Event Stream Manager (for example, when you import content from SAS Event Stream Manager directly into your project version in SAS Event Stream Manager).

You can view a project’s major versions and minor versions in the version hierarchy on the **Versioning** page.

Publish a Version of a Project

1 On the **Projects** page, right-click the relevant project and select **Open Project**.

SAS Event Stream Processing Studio Modeler appears.

Note: To create a new version of a project, the version must exist in a valid state.

2 Click  .

The **Versioning** page appears. This page contains a version hierarchy that displays the versions of the project that you are working on.

3 Click  .

The Publish — Version window appears.

a In the **Version notes** field, enter any notes that relate to the version of the project that you want to publish. This enables you to maintain a record of a project's version history.

Note: You cannot modify the version notes of a project version that has already been published.

b Click **OK**.

The **Versioning** page displays your published project version in the version hierarchy.

4 To view information relating to the project version that you published, select the relevant version in the version hierarchy on the left.

When you publish a project version, the version's XML code is updated to display its unique ID number. The project's ID number, major version number, and minor version number are specified in the version's XML code as metadata. For more information, see "Overview" on page 8.

View a Published Version of a Project

1 On the **Projects** page, right-click the relevant project and select **Open Project**.

SAS Event Stream Processing Studio Modeler appears.

2 Click  .

The **Versioning** page appears. This page displays a version hierarchy containing the current and previous versions of the project.

3 In the version hierarchy on the left, select the version of the project that you want to view in SAS Event Stream Processing Studio Modeler.

4 Click  .

The published version is displayed in Read-Only mode.

Note: You cannot make changes to a version of a project that has already been published. If you want to make more changes to a project, a new working copy is made available for you to edit in SAS Event Stream Processing Studio.

Revert to a Previous Version

1 On the **Projects** page, right-click the relevant project and select **Open Project**.

SAS Event Stream Processing Studio Modeler appears.

2 Click  .

The **Versioning** page appears. This page displays a version hierarchy containing the current and previous versions of the project.

3 In the version hierarchy on the left, select the version of the project that you want to revert to.

4 Click  .

The Revert to Version window appears.

5 Click **Yes** to confirm the reversion.

The working version of the project reverts to the published version that you selected in the version hierarchy. You cannot undo this operation.

Download a Previously Published Version

1 On the **Projects** page, right-click the relevant project and select **Open Project**.

SAS Event Stream Processing Studio Modeler appears.

2 Click  .

The **Versioning** page appears. This page displays a version hierarchy containing the current and previous versions of the project.

3 In the version hierarchy on the left, select the version of the project that you want to download.

4 Click  .

The project version downloads to your computer.

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

Example: Processing Trades

Prepare the Example Files for Use

1 If you have not already done so, download the SAS Event Stream Processing examples package from https://support.sas.com/downloads/package.htm?pid=2421 and extract its contents on your computer.

2 In the examples package, navigate to the `/xml/trades_xml` directory.

3 Save the trades.csv and traders.csv files on the server on which SAS Event Stream Processing is running. It is recommended that you follow the steps to create the model. However, if you want to run the model without following the steps to create it, upload the trades.xml file to SAS Event Stream Processing Studio. For more information, see "Upload a Project" on page 11.

For more information about these files, see "Overview" on page 47.

Overview

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:

- events about securities transactions are streamed to a Source window called Trades
- receives information about traders using a Source window called Traders
- identifies large trades using a Filter window called LargeTrades
- retains a list of the large trades using a Copy window called LargeTradesCopy
- 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
- retains a list of the large trades with their total cost using a Copy window called TotalCostCopy
- aggregates the large trades by security using an Aggregate window called BySecurity

This example uses three files listed below:

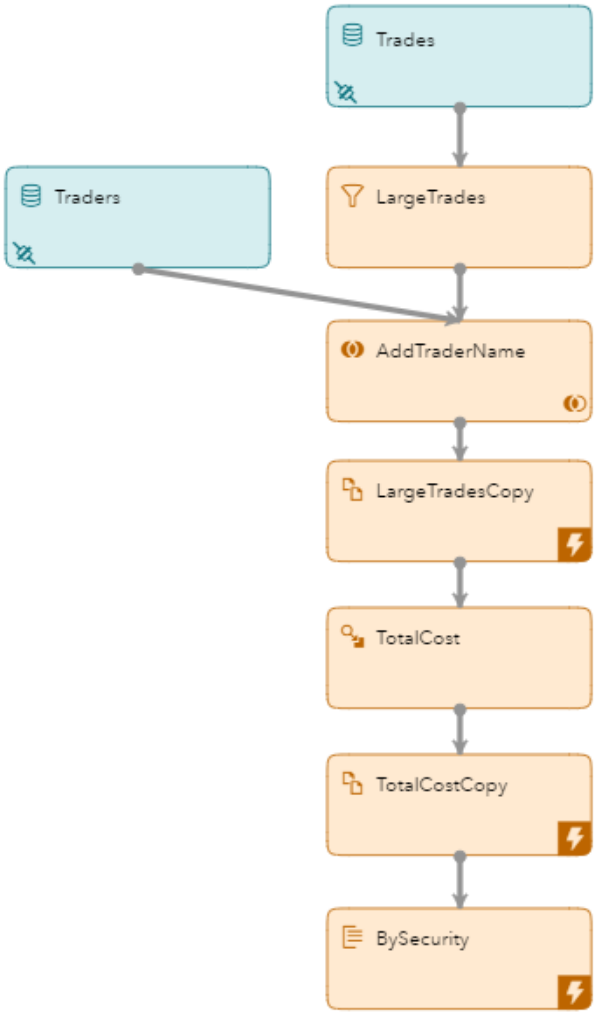
- The XML file (trades.xml) associated with this example.
- trades.csv is an input file. This file contains events relating to securities transactions.
- traders.csv is an input file. This file contains events relating to the traders involved in the securities transactions.

Project Details

This project contains eight windows:

- Trades is a Source window. This is where the securities from the trades.csv file enter the model.
- Traders is a Source window. This is where the trader names from the traders.csv file enter the model.
- LargeTrades is a Filter window. This window filters out all trades not in the specified range.
- LargeTradesCopy is a Copy window. This window retains a list of all large trades.
- AddTraderName is a Join window. This window combines the large trades with their corresponding trader names.
- TotalCost is a Compute window. This window shows the total cost of each transaction. You can use this information to identify high-value transactions.
- TotalCostCopy is a Copy window. This window retains a list of the total costs of each transaction in accordance with the retention policy
- BySecurity is an Aggregate window. This window shows all the inserts, deletes, and update blocks for the large trades.

Figure 23 *Diagram of the Processing Trades Project*



Example Steps

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 `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 ESP servers configured, you are prompted to decide whether you want to configure an ESP server now. If you are using a Kubernetes cluster, an ESP

server is automatically created for you when you run the project in test mode. Therefore, you do not need to manually create an ESP server.

Note: It is assumed that you do not have any ESP servers configured. If you already have ESP servers configured or you are going to run the project on an ESP server in a cluster, go to step 5.

3 Click **Yes** to configure an ESP server now or click **No** to skip the ESP server creation process.

If you selected **Yes**, the ESP Server Properties window appears.

4 If you selected **No**, skip this step. If you selected **Yes** to configure an ESP server now, do the following:

a In the **Name** field, enter or update the name that identifies the ESP server.

b In the **Host** field, enter or update the ESP server's host name or IP address.

c In the **HTTP port** field, enter or update the ESP server's administration port number.

d If required, in the **Description** field, enter or update the description of the ESP server.

e If required, in the **Tags** field, enter or update any keywords that describe the ESP server and then press Enter.

f If required, click **Edit** to change the setting for the **Authentication** field:

The Authentication window appears.

- **None**: This is the default option.
- **Kerberos**: This option is relevant only if the ESP server is configured to require authentication using Kerberos.
- **OAuth token**: This option is relevant only if the ESP server is configured to require authentication. If you select this option, an additional field appears where you must enter the OAuth token.
- **Username and password**: This option is relevant only if the ESP server is configured to require authentication using a user name and password (SASLogon Services). If you select this option, additional fields appear where you must enter or confirm the user name and password.

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

h If required, select the **Enable server logging** check box to enable logging on the ESP server.

i If required, in the **Number of messages to retain** field, change the default number of messages that are retained by the ESP server log. The default is 10,000 messages.

j Click **OK**.

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

a Expand **Attributes**.

b 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 securities transactions.

8 Specify a name for the Source window: In the right pane, in the **Name** field, change the default name to `Trades`.

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	Stamp

d Click **OK**.

10 Configure the Trades window to stream events from a file called trades.csv that contains securities transactions. To add a connector to this CSV file:

a In the right pane, click  .

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

c Click  .

The Connector Configuration window appears.

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

e In the **Fsname** field, enter the full path to the CSV file.

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 and event type for the Trades window:

a Expand **State and Event Type**.

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

c Select the **Accept only "Insert" events** check box.

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 for the Source window: In the right pane, in the **Name** field, change the default name to `Traders`.

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. To add a connector to this CSV file:

a In the right pane, click  .

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

c Click  .

The Connector Configuration window appears.

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

e In the **Fsname** field, enter the full path to the CSV file.

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

g Click **OK**.

17 Specify a state and event type for the Traders window:

a Expand **State and Event Type**.

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

c Select the **Accept only "Insert" events** check box.

18 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.

19 Click the newly created Filter window on the workspace.

The right pane displays the Filter window's properties.

20 Specify a name for the Filter window: In the **Name** field, change the default name to `LargeTrades`.

21 Specify a filter expression for the LargeTrades window:

a Expand **Filter**.

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

c Click  to validate the expression.

A message appears, informing you of the expression's validity.

Note: To validate your expression successfully, a working ESP server must be available. If you are using only a Kubernetes environment, skip this step as expression validation is not supported for a project running on a cluster.

d Confirm that the expression is valid.

e Collapse **Filter**.

22 Configure the LargeTrades window's state:

a Expand **State**.

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

23 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 color changes 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.

24 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.

25 Specify a name and description for the Join window: In the right pane, in the **Name** field, change the default name to `AddTraderName`.

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

The AddTraderName window now accepts trades from the LargeTrades window.

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

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

28 Click the AddTraderName window on the workspace.

The right pane displays the AddTraderName window's properties.

29 Confirm the AddTraderName window's configuration settings:

a 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.

b In the **Output field calculation method** field, confirm that **Select fields** is selected from the drop-down list.

Selecting the **Select fields** option ensures that, as new input events arrive, join non-key fields are calculated using a join selection string. This selection string is a one-to-one mapping of input fields to join fields.

c Collapse **Settings**.

30 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**.

31 Confirm the AddTraderName window's join criteria:

a Expand **Join Criteria** if it is not already expanded.

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

c Select the **Set the "no-regenerates" option** check box.

d Collapse **Join Criteria**.

32 Configure the AddTraderName window's state:

a Expand **State**.

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

33 Specify a schema for the AddTraderName window:

a In the right pane, click  .

b Click  .

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

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

The Edit Output Schema window displays the fields that you selected.

c Click **OK**.

You are returned to the Edit Output Schema – Non-Key Fields window.

d Click **OK**.

34 In the right pane, click  .

35 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.

36 Specify a name and description for the Copy window. In the right pane, in the **Name** field, change the default name to `LargeTradesCopy`.

37 Configure the LargeTradesCopy window's retention properties:

a Click the LargeTradesCopy window in the workspace.

b Expand **Retention**.

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

d In the **Time limit** field, enter `30` and select **Minutes** from the drop-down list.

38 Configure the output rules for the LargeTradesCopy window:

a Expand **Output Rules**.

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

39 Connect the AddTraderName window to the LargeTradesCopy window with an edge.

The LargeTradesCopy window now accepts trades from the AddTraderName window.

40 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.

41 Specify a name for the Compute window: In the **Name** field, change the default name to `TotalCost`.

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

The TotalCost window now accepts trades from the LargeTradesCopy window.

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

44 Configure the TotalCost window's state:

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

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

45 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 the 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**.

If you copied schema fields you previously created, you must still manually enter all new fields and their values.

Key	Field Name	Type	Expression
Y	tradeID	String	(not applicable)
N	security	String	security
N	quantity	Int32	quantity

Key	Field Name	Type	Expression
N	price	Double	price
N	totalCost	Double	price*quantity
N	traderID	Int64	traderID
N	time	Stamp	time
N	name	String	name

d Click **OK**.

46 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.

47 Specify a name and description for the Copy window. In the right pane, in the **Name** field, change the default name to `TotalCostCopy`.

48 Configure the TotalCostCopy window's retention properties:

a Click the TotalCostCopy window in the workspace.

b Expand **Retention**.

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

d In the **Time limit** field, enter `30` and select **Minutes** from the drop-down list.

49 Configure the output rules for the TotalCostCopy window:

a Expand **Output Rules**.

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

50 Connect the TotalCost window to the TotalCostCopy window with an edge.

The TotalCostCopy window now accepts trades from the TotalCost window.

51 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.

52 In the right pane, click  .

53 Specify a name for the Aggregate window: In the **Name** field, change the default name to `BySecurity`.

54 Connect the TotalCostCopy window to the BySecurity window with an edge.

The BySecurity window now accepts trades from the TotalCostCopy window.

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

56 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**.

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

58 Click Enter Test Mode.

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

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

60 Click Run Test.

The results for each window appear on separate tabs:

- The **Trades** tab lists the 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 **LargeTradesCopy** tab lists the retained large trades in accordance with the retention policy
- 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 **TotalCostCopy** tab lists the retained total cost of each transaction in accordance with the retention policy
- The **BySecurity** tab shows all the inserts, deletes, and update blocks for the large trades. 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.

61 To stop the test, click ■ Stop.

The project stops and then unloads from the ESP server.

Example: Streaming Analytics with Scoring and Training

Prepare the Example Files for Use

1 If you have not already done so, download the SAS Event Stream Processing examples package from https://support.sas.com/downloads/package.htm?pid=2421 and extract its contents on your computer.

2 In the examples package, navigate to the `/analytics/analytics_kmeans` directory.

3 Save the input.csv file on the server on which SAS Event Stream Processing is running. If you want to run the model without following the steps to create it, upload the model.xml file to SAS Event Stream Processing Studio. For more information, see “Upload a Project” on page 11.

For more information about these files, see “Overview” on page 58.

Overview

This example demonstrates the use of the machine learning algorithm *k-means*, which is often used for cluster analysis in data mining. The algorithm assigns data points to their nearest cluster centroid. Each cluster centroid is then recomputed based on the average of the cluster’s data points. In *k-means* clustering, the input event is augmented with a cluster number. This indicates the cluster that the observation falls into.

This example uses two files:

- The XML file (model.xml) associated with this example.
- input.csv is an input file. This file contains the events to be scored.

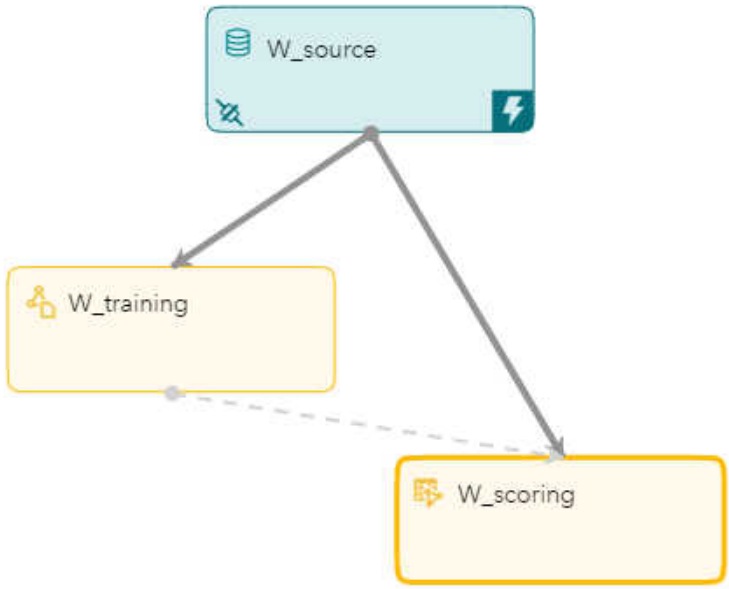
Project Details

This project contains three windows:

- w_source is a Source window. This is where events from the input.csv file enters the model to be scored.

- w_training is a Train window. This window generates and periodically updates the *k-means* model.
- w_scoring is a Score window. This is where the events are scored.

Figure 24 *Diagram of the Streaming Analytics Model with Scoring and Training*



Example Steps

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 `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 ESP servers configured, you are prompted to decide whether you want to configure an ESP server now. If you are using a Kubernetes cluster, an ESP server is automatically created for you when you run the project in test mode. Therefore you do not need to manually create an ESP server.

Note: It is assumed that you do not have any ESP servers configured. If you already have ESP servers configured or you are going to run the project on an ESP server in a cluster, go to step 5.

3 Click **Yes** to configure an ESP server now or click **No** to skip the ESP server creation process.

If you selected **Yes**, the ESP Server Properties window appears.

4 If you selected **No**, skip this step. If you selected **Yes** to configure an ESP server now, do the following:

   a In the **Name** field, enter or update the name that identifies the ESP server.

   b In the **Host** field, enter or update the ESP server's host name or IP address.

   c In the **HTTP port** field, enter or update the ESP server's administration port number.

   d If required, in the **Description** field, enter or update the description of the ESP server.

   e If required, in the **Tags** field, enter or update any keywords that describe the ESP server and then press Enter.

   f If required, click **Edit** to change the setting for the **Authentication** field:

      The Authentication window appears.

      - **None**: This is the default option.
      - **Kerberos**: This option is relevant only if the ESP server is configured to require authentication using Kerberos.
      - **OAuth token**: This option is relevant only if the ESP server is configured to require authentication. If you select this option, an additional field appears where you must enter the OAuth token.
      - **Username and password**: This option is relevant only if the ESP server is configured to require authentication using a user name and password (SASLogon Services). If you select this option, additional fields appear where you must enter or confirm the user name and password.

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

   h If required, select the **Enable server logging** check box to enable logging on the ESP server.

   i If required, in the **Number of messages to retain** field, change the default number of messages that are retained by the ESP server log. The default is 10,000 messages.

   j Click **OK**.

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 for the Source window: In the right pane, in the **Name** field, change the default name to `W_source`.

8 Configure W_source window's event type:

   a In the right pane, expand **State and Event Type**.

   b Select the **Accept only "Insert" events** check box.

9 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**.

10 The W_source window will stream events from a file called input.csv that contains example data. To add a connector to this CSV file:

a In the right pane, click  .

b If it is not already selected, click the W_source window to select it.

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

The Connector Configuration window appears.

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

e In the **Fsname** field, enter full the path to the CSV file.

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**.

11 Configure an output rule for the W_source window:

a Expand **Output Rules**.

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

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

This window uses the *k-means* algorithm to periodically generate a new clustering model.

The right pane displays the Train window's properties.

13 Specify a name for the Train window: In the right pane, in the **Name** field, change the default name to `W_training`.

14 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 color changes 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.

15 If it is not already selected, click the W_training window on the workspace to select it.

16 If it is not already expanded, expand **Settings**.

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

18 Expand **Parameters**:

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

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

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

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

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

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

g In the **velocity** field, enter `5` to specify the number of events that arrive at a single timestamp.

h 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`.

i Collapse **Parameters**.

j Expand **Input Map**.

k In the **Field** column in the table, click the row in the table twice and select **x_c** and **y_c** from the drop-down list. These variables are to be used in the clustering.

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 Enter a name for the Score window: 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 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

22 Click **OK**.

23 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.

24 Configure the settings for the W_scoring window:

a Click the W_scoring window on the workspace.

b Expand **Settings** if it is not already expanded.

c 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**.

d Expand **KMEANS**.

e Configure an input map:

i Expand **Input Map**.

ii In the **Field** column in the table, click the row in the table twice and select **x_c** and **y_c** from the drop-down list. These variables are to be used in the clustering.

iii Collapse **Input Map**.

f 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**.

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

26 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 **Enable trace server logging for this query** field, select **W_scoring** and **W_training**.

27 Click .

28 Click Enter Test Mode.

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

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

30 Click Run Test.

The results for each window appear in separate tabs:

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

31 To stop the test, click Stop.

Example: Using a Geofence

Prepare the Example Files for Use

1. If you have not already done so, download the SAS Event Stream Processing examples package from https://support.sas.com/downloads/package.htm?pid=2421 and extract its contents on your computer.
2. In the examples package, navigate to the `/xml/geofence2` directory.
3. Save the anpr.csv, infrastructure.csv, and wantedvehicle.csv files on the server on which SAS Event Stream Processing is running. If you want to run the model without following the steps to create it, upload the ANPR.xml file to SAS Event Stream Processing Studio. For more information, see “Upload a Project” on page 11.

   For more information about these files, see “Overview” on page 65.

Overview

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 that are 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

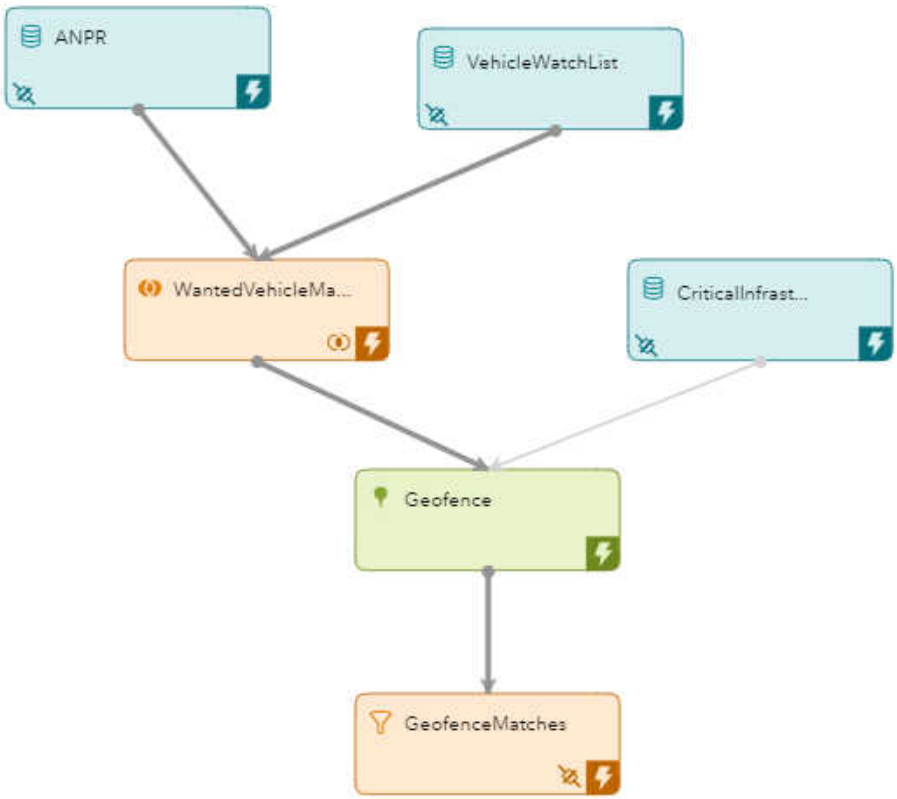
Project Details

This project contains six windows:

- The ANPR window is a Source window. This is where a list of all vehicles within close proximity of critical infrastructure sites from the anpr.csv file enter the model.
- The VehicleWatchList window is a Source window. This is where a list of all vehicles on the vehicle watch list from the wantedvehicle.csv file enter the model.
- The CriticalInfrastructure window is a Source window. This is where a list of sites that contain critical infrastructure from the infrastructure.csv file enter the model.
- The WantedVehicleMatch window is a Join window. This is where a list of all vehicles found within close proximity of critical infrastructure sites, and a list of all wanted vehicles are merged into one list.
- The Geofence window is a Geofence window. This is where geofencing information that relates to the matched vehicles enter the model.

- The GeofenceMatches window is a Filter window. This is where the geofencing information is filtered.

Figure 25 *Diagram of the Geofence Model*



Example Steps

To complete this example, follow the steps below:

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

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 ESP servers configured, you are prompted to decide whether you want to configure an ESP server now. If you are using a Kubernetes cluster, an ESP server is automatically created for you when you run the project in test mode. Therefore you do not need to manually create an ESP server.

Note: It is assumed that you do not have any ESP servers configured. If you already have ESP servers configured or you are going to run the project on an ESP server in a cluster, skip to step 5.

3 Click **Yes** to configure an ESP server now or click **No** to skip the ESP server creation process.

If you selected **Yes**, the ESP Server Properties window appears.

4 If you selected **No**, skip this step. If you selected **Yes** to configure an ESP server now, do the following:

a In the **Name** field, enter or update the name that identifies the ESP server.

b In the **Host** field, enter or update the ESP server's host name or IP address.

c In the **HTTP port** field, enter or update the ESP server's administration port number.

d If required, in the **Description** field, enter or update the description of the ESP server.

e If required, in the **Tags** field, enter or update any keywords that describe the ESP server and then press Enter.

f If required, click **Edit** to change the setting for the **Authentication** field:

The Authentication window appears.

- **None**: This is the default option.
- **Kerberos**: This option is relevant only if the ESP server is configured to require authentication using Kerberos.
- **OAuth token**: This option is relevant only if the ESP server is configured to require authentication. If you select this option, an additional field appears where you must enter the OAuth token.
- **Username and password**: This option is relevant only if the ESP server is configured to require authentication using a user name and password (SASLogon Services). If you select this option, additional fields appear where you must enter or confirm the user name and password.

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

h If required, select the **Enable server logging** check box to enable logging on the ESP server.

i If required, in the **Number of messages to retain** field, change the default number of messages that are retained by the ESP server log. The default is 10,000 messages.

j Click **OK**.

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 for the Source window: In the right pane, in the **Name** field, change the default name to `ANPR`.

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	Stamp
Y	pkey	String

d Click **OK**.

9 The ANPR window streams a list of vehicles from a file called anpr.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 Connector Configuration window appears.

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

d In the **Fsname** field, enter the full path to the CSV file.

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 for the Source window: In the right pane, in the **Name** field, change the default name to `VehicleWatchList`.

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 streams a list of wanted vehicles from a file called wantedvehicle.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 Connector Configuration window appears.

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

d In the **Fsname** field, enter the full path to the CSV file.

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 for the Join window: In the right pane, in the **Name** field, change the default name to `WantedVehicleMatch`.

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 color changes to white.
- b Click the white anchor point, hold the left 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 24. 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 Examine 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, confirm that **Select fields** is selected from the drop-down list.

  As a result of choosing the **Select fields** option, as new input events arrive, join non-key fields are calculated using a join selection string. This selection string is a one-to-one mapping of input fields to join fields.
- d Collapse **Settings**.

20 Configure the WantedVehicleMatch window's join criteria:

- a If it is not already expanded, 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 twice, and select **vrm** from the drop-down list.

d Click the cell in the Right: VehicleWatchList column twice, 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  .

The Edit Output Schema window appears.

Use this window to configure the fields as shown in the following table. As 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
ANPR	vrm	String
ANPR	lat	Double
ANPR	long	Double
ANPR	date	Stamp

The Edit 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`.

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 streams a list of sites that contain critical infrastructure from a file called infrastructure.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**.

c Click  .

The Connector Configuration window appears.

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

e In the **Fsname** field, enter the full path to the CSV file.

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

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 **X coordinate** field, select **long** from the drop-down list.

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

d In the **Default lookup distance (meters)** field, enter `20`.

e Collapse **Positions**.

34 Configure the Geofence window's geometric settings:

a Expand **Geometries**.

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

c In the table, in the **Y coordinate** 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 for the Filter window: In the **Name** field, change the default name to `GeofenceMatches`.

39 Configure a subscribe connector for the GeofenceMatches window:

a Expand **Subscriber Connectors**.

b Click  .

The Connector Configuration window appears.

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

d Select the **Snapshot** check box.

e In the **Fsname** field, enter the full path to the output file: **result.out**.

f In the **Fstype** field, enter `csv`.

g Click **OK**.

40 Specify a filter expression for the GeofenceMatches window:

a Expand **Filter**.

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

c Click  to validate the expression.

A message appears, informing you of the expression's validity.

Note: To validate your expression successfully, a working ESP server must be available. If you are using only a Kubernetes environment, skip this step as expression validation is not supported for a project running on a cluster.

d Confirm that the expression is valid.

e Collapse **Filter**.

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

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

Row	Controlling Group	Dependent Group
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"/>`

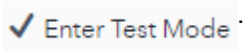
43 Configure your model's threading level:

a In the right pane, click  .

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

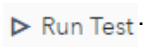
c In the **Threads** field, enter `8`.

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

45 Click  .

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

46 In the **ESP Server** drop-down list, confirm that the ESP server on which you want to test the model is selected. If the appropriate ESP server is not selected, select it from the drop-down list.

47 Click  .

The results for each window appear 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 combines a list of all vehicles found within close proximity of critical infrastructure sites with a list of all wanted vehicles.
- The **Geofence** tab lists the geofencing information that relates to the matched vehicles
- The **CriticalInfrastructure** tab lists sites that contain critical infrastructure
- The **GeofenceMatches** tab shows any 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

Prepare the Example Files for Use

1 Download the esp-6.1.TextAnalyticsExample.zip file from `ftp://ftp.sas.com/techsup/download/esp/esp-6.1.TextAnalyticsExample.zip`.

2 Navigate to the esp-6.1.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 Save the citng_en-ne.li, IPTC.mco, and textanalytics.csv files on the server on which SAS Event Stream Processing is running. If you want to run the model without following the steps to create it, upload the text_analytics.xml file to SAS Event Stream Processing Studio. For more information, see “Upload a Project” on page 11.

For more information about these files, see “Overview” on page 77.

Overview

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 that is 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.

This example uses five files listed below:

- esp-6.1.TextAnalyticsExample.zip contains the files that you need to complete this example. You can download the esp-6.1.TextAnalyticsExample.zip file from `ftp://ftp.sas.com/techsup/download/esp/esp-6.1.TextAnalyticsExample.zip`.
- The XML file (text_analytics.xml) associated with this example.
- textanalytics.csv is an input file. This file contains data to be analyzed, categorized, and sorted in context.
- citng_en-new.li is a LITI file. This file contains the rules from which contextual information is generated.

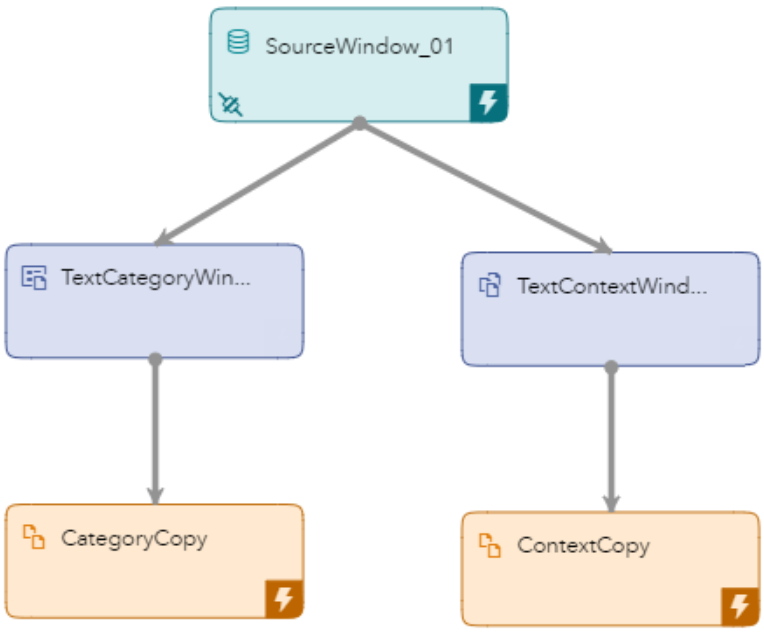
- IPTC.mco is an MCO file. This file contains the rules from which the information in the input is analyzed and categorized.

Project Details

This project contains five windows:

- SourceWindow_01 is a Source window. This is where data from the textanalytics.csv file enters the model. This data is made available for analysis.
- TextCategoryWindow_01 is a Text Category window. This window displays information in categorized format.
- TextContextWindow_01 is a Text Context window. This window displays information in contextual format.
- CategoryCopy is a Copy window. This window displays categorized events with a retention policy of 30 seconds.
- ContextCopy is a Copy window. This window displays contextual events with a retention policy of 30 seconds.

Figure 26 *Diagram of the Text Analytics Model*



Example Steps

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

The New Project window appears.

2 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 ESP servers configured, you are prompted to decide whether you want to configure an ESP server now. If you are using a Kubernetes cluster, an ESP server is automatically created for you and you do not need to manually create an ESP server.

   Note: It is assumed that you do not have any ESP servers configured. If you already have ESP servers configured or you are going to run the project on an ESP server in a cluster, go to step 7.

3 Click **Yes** to configure an ESP server now or click **No** skip the ESP server creation process.

   If you selected **Yes**, the ESP Server Properties window appears.

4 If you selected **No**, skip this step. If you selected **Yes** to configure an ESP server now, do the following:

   a In the **Name** field, enter or update the name that identifies the ESP server.

   b In the **Host** field, enter or update the ESP server's host name or IP address.

   c In the **HTTP port** field, enter or update the ESP server's administration port number.

   d If required, in the **Description** field, enter or update the description of the ESP server.

   e If required, in the **Tags** field, enter or update any keywords that describe the ESP server and then press Enter.

   f If required, click **Edit** to change the setting for the **Authentication** field:

   The Authentication window appears.

   - **None**: This is the default option.
   - **Kerberos**: This option is relevant only if the ESP server is configured to require authentication using Kerberos.
   - **OAuth token**: This option is relevant only if the ESP server is configured to require authentication. If you select this option, an additional field appears where you must enter the OAuth token.
   - **Username and password**: This option is relevant only if the ESP server is configured to require authentication using a user name and password (SASLogon Services). If you select this option, additional fields appear where you must enter or confirm the user name and password.

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

   h If required, select the **Enable server logging** check box to enable logging on the ESP server.

   i If required, in the **Number of messages to retain** field, change the default number of messages that are retained by the ESP server log. The default is 10,000 messages.

   j Click **OK**.

5 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`.

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 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`.

8 Configure the SourceWindow_01 window's event type:

   a In the right pane, expand **State and Event Type**.

   b Select the **Accept only "Insert" events** check box.

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

9 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	Int64
N	tstamp	Date
N	msg	String

   d Click **OK**.

10 The SourceWindow_01 window streams 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 Connector Configuration window appears.

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

d In the **Fsname** field, enter the full path to the CSV file.

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

f Configure the DataIn connector's properties:

i Click **All properties**.

The All Properties — DataIn 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**.

11 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.

12 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.`

13 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 color changes 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.

14 Click the TextCategoryWindow window on the workspace.

15 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 path** field, enter the full path to your IPTC.mco file.

16 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.

17 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 30 seconds.`

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

19 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.

20 Configure the CategoryCopy window's retention properties:

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

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

c In the **Time limit** field, enter `30` and confirm that **Seconds** is selected from the drop-down list.

d Collapse **Retention**.

21 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.

22 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.`

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

24 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 paths** field, click  .

A new row is created in the table.

e Click the new row and enter the full path of your LITI file.

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

25 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.

26 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 30 seconds.`

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

28 Configure the ContextCopy window's state and index:

a Click the ContextCopy window on the workspace.

b Expand **State**.

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

29 Configure the ContextCopy window's retention properties:

a Click the ContextCopy window in the workspace.

b Expand **Retention**.

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

d In the **Time limit** field, enter `30` and confirm that **Seconds** is selected from the drop-down list.

30 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**.

31 Click .

32 Click ✓ Enter Test Mode.

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

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

34 Click ▷ Run Test.

The results for each window appear on separate tabs:

- The **CategoryCopy** tab displays categorized events with a retention policy of 30 seconds
- The **ContextCopy** tab displays contextual events with a retention policy of 30 seconds
- The **TextContextWindow** tab displays information in contextual format
- The **SourceWindow_01** tab displays the processed event stream
- The **TextCategoryWindow** tab displays information in categorized format

35 To stop the test, click ■ Stop.

The project stops and then unloads from the ESP server.

Example: Unifying Multiple Input Streams

Prepare the Example Files for Use

1. If you have not already done so, download the SAS Event Stream Processing examples package from https://support.sas.com/downloads/package.htm?pid=2421 and extract its contents on your computer.
2. In the examples package, navigate to the `/xml/union_xml` directory.
3. Save the input_sw_01_1.csv, input_sw_01_2.csv, and input_sw_02.csv files on the server on which SAS Event Stream Processing is running. If you want to run the model without following the steps to create it, upload the model.xml file to SAS Event Stream Processing Studio. For more information, see “Upload a Project” on page 11.

   For more information about these files, see “Overview” on page 84.

Overview

By following this example, you learn how to create a model that merges one or more streams with the same schema. The model contains two Source windows and one Union window. The two Source windows stream multiple lists of stock prices to the Union window. The Union window is configured with strict enforcement, that is, the key merge from each window must semantically merge cleanly.

This example uses five files listed below:

- The XML file (model.xml) associated with this example.
- input_sw_01_1.csv is an input file. This file contains a list of stock prices.
- input_sw_01_2.csv is an input file. This file contains a list of stock prices.
- input_sw_02.csv is an input file. This file contains a list of stock prices.
- output.csv is an output file. When you run the model, the unified list of stock prices is written to the output.csv file.

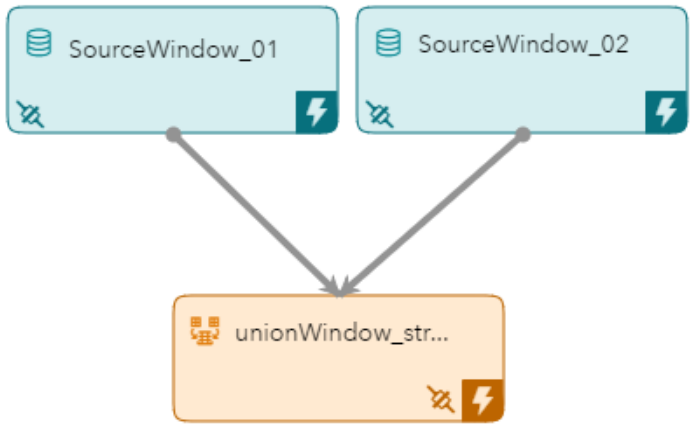
Project Details

This project contains three windows:

- The sourceWindow_01 window is a Source window. This is where a list of stock prices from the input_sw_01_1.csv file and a list of stock prices from the input_sw_01_2.csv file enter the model.

- The sourceWindow_02 window is a Source window. This is where a list of stock prices from the input_sw_02.csv file enters the model.
- The unionWindow_strict window is a Union window. This is where the multiple lists of stock prices are merged into one list. This window also writes the results of the merge to the output.csv file.

Figure 27 *Diagram of the Unifying Multiple Input Streams Project*



Example Steps

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

The New Project window appears.

2 In the New Project window, do the following:

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

b In the **Description** field, enter a description. Here is an example: `This model merges three event streams of stock prices using a Union window.`

c Click **OK**.

If you do not currently have any ESP servers configured, you are prompted to decide whether you want to configure an ESP server now.If you are using a Kubernetes cluster, an ESP server is automatically created for you when you run the project in test mode. Therefore you do not need to manually create an ESP server.

Note: It is assumed that you do not have any ESP servers configured. If you already have ESP servers configured or you are going to run the project on an ESP server in a cluster, go to step 5.

3 Click **Yes** to configure an ESP server now or click **No** to skip the ESP server creation process.

If you selected **Yes**, the ESP Server Properties window appears.

4 If you selected **No**, skip this step. If you selected **Yes** to configure an ESP server now, do the following:

a In the **Name** field, enter or update the name that identifies the ESP server.

b In the **Host** field, enter or update the ESP server's host name or IP address.

c In the **HTTP port** field, enter or update the ESP server's administration port number.

d If required, in the **Description** field, enter or update the description of the ESP server.

e If required, in the **Tags** field, enter or update any keywords that describe the ESP server and then press Enter.

f If required, click **Edit** to change the setting for the **Authentication** field:

The Authentication window appears.

- **None**: This is the default option.
- **Kerberos**: This option is relevant only if the ESP server is configured to require authentication using Kerberos.
- **OAuth token**: This option is relevant only if the ESP server is configured to require authentication. If you select this option, an additional field appears where you must enter the OAuth token.
- **Username and password**: This option is relevant only if the ESP server is configured to require authentication using a user name and password (SASLogon Services). If you select this option, additional fields appear where you must enter or confirm the user name and password.

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

h If required, select the **Enable server logging** check box to enable logging on the ESP server.

i If required, in the **Number of messages to retain** field, change the default number of messages that are retained by the ESP server log. The default is 10,000 messages.

j Click **OK**.

5 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 `3`.

6 Configure the project's continuous query:

a Click  .

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

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.

8 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 defines an event stream. All event streams must enter continuous queries by being published or injected into a source window.`

9 Specify an output schema for the SourceWindow_01 window:

a On the right toolbar, 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	symbol	String
N	price	Double

d Click **OK**.

10 The SourceWindow_01 window streams a list of vehicles from a file called input_sw_01_1.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 Connector Configuration window appears.

c In the **Name** field, replace the default value with `pub_sw_01_1`.

d In the **Fsname** field, enter the full path to the CSV file.

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

f Configure the pub_sw_01_1 connector's properties:

i Click **All properties**.

The All Properties – pub_sw_01_1 window appears.

ii Enter `3` in the **Value** field of the **blocksize** property.

iii Click **OK**.

g Click **OK**.

11 The SourceWindow_01 window streams a list of vehicles from a file called input_sw_01_2.csv that contains example data. To add a connector to this CSV file:

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

The Connector Configuration window appears.

b In the **Name** field, replace the default value with `pub_sw_01_2`.

c In the **Fsname** field, enter the full path to the CSV file.

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

e Configure the pub_sw_01_2 connector's properties:

i Click **All properties**.

The All Properties – pub_sw_01_2 window appears.

ii Enter `3` in the **Value** field of the **blocksize** property.

iii Click **OK**.

f Click **OK**.

12 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.

13 Specify a name and description for the Source window:

a In the right pane, in the **Name** field, change the default name to `SourceWindow_02`.

b In the **Description** field, enter `This window defines an event stream. All event streams must enter continuous queries by being published or injected into a source window`.

14 Specify an output schema for the SourceWindow_02 window:

a On the right toolbar, 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	symbol	String
N	price	Double

d Click **OK**.

15 Configure the SourceWindow_02 window's state and index:

a On the right toolbar, click  .

b Expand **State and Event Type**.

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

16 The SourceWindow_02 window streams a list of vehicles from a file called input_sw_02.csv that contains example data. To add a connector to this CSV file:

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

The Connector Configuration window appears.

b In the **Name** field, replace the default value with `pub_sw_02`.

c In the **Fsname** field, enter the full path to the CSV file.

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

e Configure the pub_sw_02 connector's properties:

i Click **All properties**.

The All Properties – pub_sw_02 window appears.

ii Enter `3` in the **Value** field of the **blocksize** property.

iii Click **OK**.

f Click **OK**.

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

The right pane displays the Union window's properties.

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

19 In the right pane, expand **Union** if necessary.

a In the **Key merging** field, confirm that **Strict** is selected.

b Collapse **Union**.

20 Configure the unionWindow_strict window's state:

a Expand **State**.

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

21 The unionWindow_strict window streams data to a CSV file (output.csv) using a subscriber connector. To configure this subscriber connector:

a Expand **Subscriber Connectors** and click  .

The Connector Configuration window appears.

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

c In the **Fsname** field, enter the full path to the CSV file.

d In the **Fstype** field, enter **csv**.

e Select the **Snapshot** check box.

f Click **OK**.

22 Connect the sourceWindow_01 window and the sourceWindow_02 window to the unionWindow_strict window with an edge:

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

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

The unionWindow_strict window now accepts values from the sourceWindow_01 window and the sourceWindow_02 window.

23 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 `CG_sub1_1`.

e Click  below the **Connectors** label.

f In the Connector column, click the newly created row and select **contquery_01/unionWindow_strict/sub_uw** 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 `CG_pub_sw_01_1`.

k Click  below the **Connectors** label.

l In the Connector column, click the newly created row and select **contquery_01/sourceWindow_01/pub_sw_01_1** 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 `CG_pub_sw_02`.

q Click  below the **Connectors** label.

r In the Connector column, click the newly created row and select **contquery_01/sourceWindow_02/pub_sw_02** 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 `CG_pub_sw_01_2`.

w Click  below the **Connectors** label.

x In the Connector column, click the newly created row and select **contquery_01/sourceWindow_01/pub_sw_01_2** 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	CG_sub_1_1	CG_pub_sw_01_1
2	CG_pub_sw_01_1	CG_pub_sw_02
3	CG_pub_sw_02	CG_pub_sw_01_2

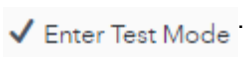
24 Configure the continuous query's trace log:

a Click  .

b Expand **Debugging**.

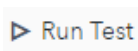
c In the **Enable trace server logging for this query** field, select **unionWindow_strict** from the drop-down list.

25 Click  .

26 Click  .

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

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

28 Click  .

The results for each window appear on separate tabs:

- The **sourceWindow_01** tab displays the first event stream
- The **sourceWindow_02** tab displays the second event stream
- The **unionWindow_strict** tab displays the unified event stream

29 To stop the test, click ■ Stop.

The project stops and is unloaded from the ESP server.

Example: Splitting Generated Events across Output Slots

Prepare the Example Files for Use

1 If you have not already done so, download the SAS Event Stream Processing examples package from https://support.sas.com/downloads/package.htm?pid=2421 and extract its contents on your computer.

2 In the examples package, navigate to the `/xml/slot_exp_xml` directory.

3 Save the input.csv file on the server on which SAS Event Stream Processing is running. If you want to run the model without following the steps to create it, upload the model.xml file to SAS Event Stream Processing Studio. For more information, see “Upload a Project” on page 11.

For more information about these files, see “Overview” on page 92.

Overview

This model enables you to send generated stock market events across a set of output slots. It contains a Source window, a Compute window, and three Copy windows. The Compute window uses an expression to determine what output slot or slots should be used for a newly generated stock market event. The Copy windows connect to the Compute window using different output slots.

Filtering events using window splitters with only one output slot can be more efficient than using multiple Filter windows. This is because the filtering is performed at the window splitter only, rather than at multiple times for each filter. For example, performing an alpha-split across a set of trades results in less data movement and data processing than performing an alpha-split across multiple Filter windows.

This example uses six files listed below:

- The XML file (model.xml) associated with this example.
- input.csv is an input file. This file contains a list of stock trades.
- compute.csv is an output file. When you run the model, the computed fields are written to the compute.csv file.
- cw_01.csv is an output file. Stock market events from slot 0 are sent to the cw_01.csv file.
- cw_02.csv is an output file. Stock market events from slot 1 are sent to the cw_02.csv file.

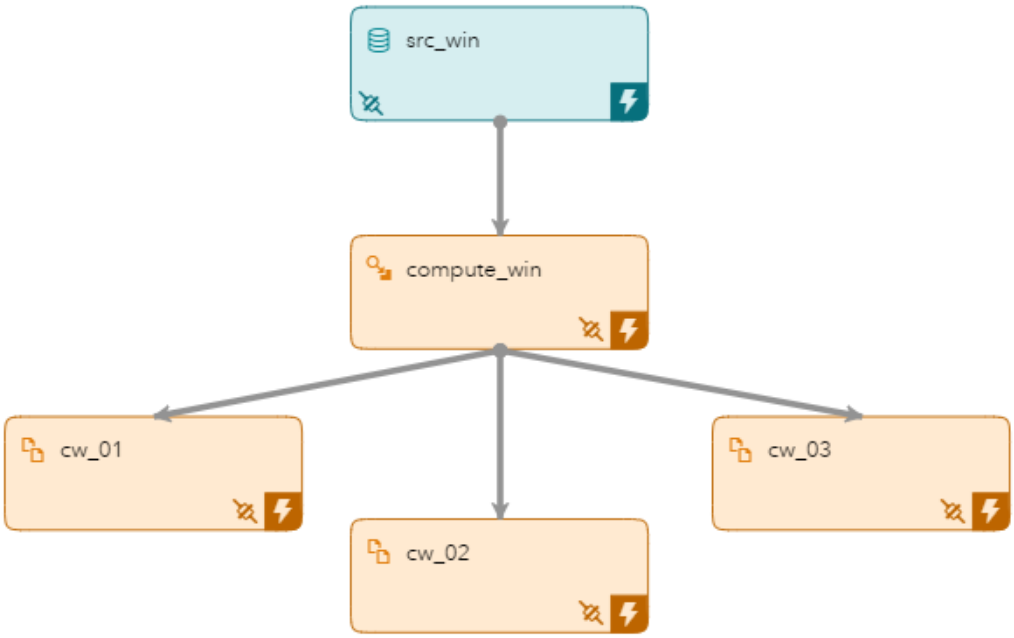
- cw_03.csv is an output file. Stock market events from slot –1 are sent to the cw_03.csv file.

Project Details

This project contains five windows:

- src_win window is a Source window. This is where a list of securities transactions from the input.csv file enter the model.
- compute_win window is a Compute window. The computed fields are listed in the compute.csv file.
- cw_01 window is a Copy window. The window writes the results of the trades allocated to slot 0 to the cw_01.csv file.
- cw_02 window is a Copy window. The window writes the results of the trades allocated to slot 1 to the cw_02.csv file.
- cw_03 window is a Copy window. The window writes the results of the trades allocated to slot –1 to the cw_03.csv file.

Figure 28 *Diagram of the Split Generated Events Across Output Slots Project*



Example Steps

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

2 In the New Project window, do the following:

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

b Click **OK**.

If no ESP servers are currently configured, you are prompted to decide whether you want to configure an ESP server now. If you are using a Kubernetes cluster, an ESP server is automatically created for you when you run the project in test mode. Therefore, you do not need to manually create an ESP server.

Note: It is assumed that no ESP servers are currently configured. If some already are or you are going to run the project on an ESP server in a cluster, go to step 5.

3 Click **Yes** to configure an ESP server now or click **No** to skip the ESP server creation process.

If you selected **Yes**, the ESP Server Properties window appears.

4 If you selected **No**, skip this step. If you selected **Yes** to configure an ESP server now, do the following:

a In the **Name** field, enter or update the name that identifies the ESP server.

b In the **Host** field, enter or update the ESP server's host name or IP address.

c In the **HTTP port** field, enter or update the ESP server's administration port number.

d If required, in the **Description** field, enter or update the description of the ESP server.

e If required, in the **Tags** field, enter or update any keywords that describe the ESP server and then press Enter.

f If required, click **Edit** to change the setting for the **Authentication** field:

The Authentication window appears.

- **None**: This is the default option.
- **Kerberos**: This option is relevant only if the ESP server is configured to require authentication using Kerberos.
- **OAuth token**: This option is relevant only if the ESP server is configured to require authentication. If you select this option, an additional field appears where you must enter the OAuth token.
- **Username and password**: This option is relevant only if the ESP server is configured to require authentication using a user name and password (SASLogon Services). If you select this option, additional fields appear where you must enter or confirm the user name and password.

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

h If required, select the **Enable server logging** check box to enable logging on the ESP server.

i If required, in the **Number of messages to retain** field, change the default number of messages that are retained by the ESP server log. The default is 10,000 messages.

j Click **OK**.

5 Configure the project's continuous query:

a Click  .

b In the **Name** field, enter `cq_01`.

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 Specify a name and description for the Source window:

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

b In the **Description** field, enter `This window receives an event stream of stock market trades.`

8 Configure the src_win window's state and event type:

a In the right pane, expand **State and Event Type**.

b Select **Stateful (pi_RBTREE)** from the **Window state and index** drop-down list.

9 Specify an output schema for the src_win window:

a In the right toolbar, 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	symbol	String
N	price	Double

d Click **OK**.

10 The src_win window streams a list of vehicles from a file called input.csv that contains example data. To add a connector to this CSV file:

a Click  .

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

The Connector Configuration window appears.

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

d In the **Fsname** field, enter the full path to the CSV file.

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

f Configure the pub connector's properties:

i Click **All properties**.

The All Properties window appears.

ii Select `true` in the **Value** field of the **transactional** property.

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

iv Click **OK**.

g Click **OK**.

11 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.

12 Specify a name and description for the Compute window:

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

b In the **Description** field, enter `This window uses expressions to calculate each field. The first field uses the expression to calculate the count. The last two fields are just passing through what is in the input window.`

13 Connect the src_win window to the compute_win window with an edge:

a Position the cursor over the anchor point at the bottom of the src_win window so that the color of the anchor point changes to white.

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

The compute_win window now accepts values from the src_win window.

14 Specify an output schema for the compute_win window:

a On the right toolbar, click  .

b Click  .

c Click  .

The Output Schema window appears.

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:

Key	Field Name	Type	Expression
Y	ID	Int32	not applicable
N	counter	Int32	counter=counter+1 return counter
N	symbol	String	symbol

Key	Field Name	Type	Expression
N	price	Double	price

e Click **OK**.

15 Configure the compute_win window's split method:

a Click .

b Expand **Advanced**.

c Select the **Split the output** check box.

d Confirm that **Expression** is selected in the **Split method** field.

e Enter `ID%2` in the **Expression** field.

f Click to validate the expression.

A message appears informing you of the expression's validity.

Note: To validate your expression successfully, a working ESP server must be available. If you are only using a Kubernetes environment, skip this step as expression validation is not supported for a project running on a cluster.

g Confirm that the expression is valid.

h Collapse **Advanced**.

16 Configure the compute_win window's compute settings:

a Expand **Compute Settings**.

b Confirm that **Expressions** is selected in the **Compute method** field.

c Select the **Include engine initialization expression** check box.

d Select **Int32** in the **Return type** drop-down list.

e In the **Expression** field, enter:

```
integer counter counter=0
```

f Click to validate the expression.

A message appears, informing you of the expression's validity.

Note: To validate your expression successfully, a working ESP server must be available. If you are only using a Kubernetes environment, skip this step as expression validation is not supported for a project running on a cluster.

g Confirm that the expression is valid.

h Collapse **Compute Settings**.

17 Configure a subscriber connector for the compute_win window:

a Expand **Subscriber Connectors** and click  .

The Connector Configuration window appears.

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

c In the **Fsname** field, enter the full path to the CSV file.

d In the **Fstype** field, enter `csv`.

e Click **OK**.

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 `cw_01`.

b In the **Description** field, enter `This Copy window connects to the Compute window using the output slot 0.`

20 Configure a subscriber connector for the cw_01 window:

a Expand **Subscriber Connectors** and click  .

The Connector Configuration window appears.

b In the **Name** field, replace the default value with `sub1`.

c In the **Fsname** field, enter the full path to the CSV file.

d Select the **Snapshot** check box.

e In the **Fstype** field, enter `csv`.

f Click **OK**.

21 Configure the cw_01 window's retention properties:

a Click the cw_01 window in the workspace.

b Expand **Retention**.

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

d In the **Time limit** field, enter `1000` and select **Seconds** from the drop-down list.

22 Connect the compute_win window to the cw_01 window with an edge:

a Position the cursor over the anchor point at the bottom of the compute_win window so that the color of the anchor point changes to white.

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

The cw_01 window now accepts values from the compute_win window.

23 Assign a slot number to the edge:

a Click the edge that connects the compute_win window with the cw_01 window.

b In the right pane, enter `0` in the **Slot** field.

24 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.

25 Specify a name and description for the Copy window:

a In the right pane, in the **Name** field, change the default name to `cw_02`.

b In the **Description** field, enter `This Copy window connects to the Compute window using the output slot 1.`

26 Configure a subscriber connector for the cw_02 window:

a Expand **Subscriber Connectors** and click  .

The Connector Configuration window appears.

b In the **Name** field, replace the default value with `sub2`.

c In the **Fsname** field, enter the full path to the CSV file.

d Select the **Snapshot** check box.

e In the **Fstype** field, enter `csv`.

f Click **OK**.

27 Configure the cw_02 window's retention properties:

a Click the cw_02 window in the workspace.

b Expand **Retention**.

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

d In the **Time limit** field, enter `1000` and select **Seconds** from the drop-down list.

28 Connect the compute_win window to the cw_02 window with an edge:

a Position the cursor over the anchor point at the bottom of the compute_win window so that the color of the anchor point changes to white.

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

The cw_02 window now accepts values from the compute_win window.

29 Assign a slot number to the edge you just created:

a Click the edge that connects the compute_win window with the cw_02 window.

b In the right pane, enter `1` in the **Slot** field.

30 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.

31 Specify a name and description for the Copy window:

a In the right pane, in the **Name** field, change the default name to `cw_03`.

b In the **Description** field, enter `This Copy window connects to the Compute window using the output slot -1.`

32 Configure a subscriber connector for the cw_03 window:

a Expand **Subscriber Connectors** and click  .

The Connector Configuration window appears.

b In the **Name** field, replace the default value with `sub3`.

c In the **Fsname** field, enter the full path to the CSV file that will contain the output.

d Select the **Snapshot** check box.

e In the **Fstype** field, enter `csv`.

f Click **OK**.

33 Configure the cw_03 window's retention properties:

a Click the cw_03 window in the workspace.

b Expand **Retention**.

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

d In the **Time limit** field, enter `1000` and select **Seconds** from the drop-down list.

34 Connect the compute_win window to the cw_03 window with an edge:

a Position the cursor over the anchor point at the bottom of the compute_win window so that the color of the anchor point changes to white.

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

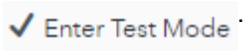
The cw_03 window now accepts values from the compute_win window.

35 Assign a slot number to the edge you just created:

a Click the edge that connects the compute_win window with the cw_03 window.

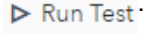
b In the right pane, enter `-1` in the **Slot** field.

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

37 Click  .

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

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

39 Click  .

The results for each window appear on separate tabs:

- The **src_win** tab lists the securities transactions
- The **compute_win** tab lists the computed fields
- The **cw_01** tab lists the trades output to slot 0

- The **cw_02** tab lists the trades output to slot 1
- The **cw_03** tab lists the trades output to slot –1

Note: If the table is empty, check that the publisher connector for the src_win window correctly points to the relevant CSV file.

40 To stop the test, click ■ Stop.

The project stops and then unloads from the ESP server.

Example: Identifying Trading Patterns in a Stock Market

Prepare the Example Files for Use

1 If you have not already done so, download the SAS Event Stream Processing examples package from https://support.sas.com/downloads/package.htm?pid=2421 and extract its contents on your computer.

2 In the examples package, navigate to the `/xml/pattern_empty_index` directory.

3 Save the 50k.csv files on the server on which SAS Event Stream Processing is running. If you want to run the model without following the steps to create it, upload the model.xml file to SAS Event Stream Processing Studio. For more information, see “Upload a Project” on page 11.

For more information about these files, see “Overview” on page 101.

Overview

This model identifies increases in a stock's price within a specific time interval. The model contains a Source window and a Pattern window. The Source window receives an event stream of stock trades from an input file that is included with this example. The Pattern window defines the events of interest to be matched. The model is stateless, that is, the index on the Source window has the type pi_EMPTY. Events are not retained in any window, and are transformed and passed through. This prevents the Pattern window from growing infinitely. The pattern defined in the Pattern window consists of the following events of interest:

- Event 1: Occurrences of the stock symbol GMTC
- Event 2: Re-occurrences of the stock symbol GMTC where the price and quantity of the stock has gone up 50% compared to event 1

- Event 3: Re-occurrences of the stock symbol GMTC where the price and quantity of the stock has gone up 50% compared to event 2

  Note: In order for the pattern to be matched, all events of interest must occur within 200 milliseconds of each other. The time at which each event occurred is specified in the trade_time field in the Source window's output schema.

This example uses three files listed below:

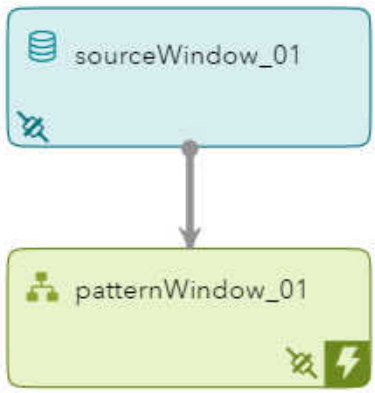
1. The XML file (model.xml) associated with this example.
2. 50k.csv is an input file. This file contains a list of stock trades.
3. output.csv is an output file. When you run the model, the matched patterns are written to the output.csv file.

Project Details

This project contains two windows:

- The sourceWindow_01 window is a Source window. This is where a list of stock trades from the 50k.csv file enter the model.
- The patternWindow_01 window is a Pattern window. This is where the stock trade patterns are matched and written to an output.csv file.

Figure 29 *Diagram of the Identifying Trading Patterns in a Stock Market Project*



Example Steps

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 `pattern_empty_index`.

b Click **OK**.

If you do not currently have any ESP servers configured, you are prompted to decide whether you want to configure an ESP server now. If you are using a Kubernetes cluster, an ESP server is automatically created for you when you run the project in test mode. Therefore you do not need to manually create an ESP server.

Note: It is assumed that you do not have any ESP servers configured. If you already have ESP servers configured or you are going to run the project on an ESP server in a cluster, go to step 5.

3 Click **Yes** to configure an ESP server now or click **No** to skip the ESP server creation process.

If you selected **Yes**, the ESP Server Properties window appears.

4 If you selected **No**, skip this step. If you selected **Yes** to configure an ESP server now, do the following:

a In the **Name** field, enter or update the name that identifies the ESP server.

b In the **Host** field, enter or update the ESP server's host name or IP address.

c In the **HTTP port** field, enter or update the ESP server's administration port number.

d If required, in the **Description** field, enter or update the description of the ESP server.

e If required, in the **Tags** field, enter or update any keywords that describe the ESP server and then press Enter.

f If required, click **Edit** to change the setting for the **Authentication** field:

The Authentication window appears.

- **None**: This is the default option.
- **Kerberos**: This option is relevant only if the ESP server is configured to require authentication using Kerberos.
- **OAuth token**: This option is relevant only if the ESP server is configured to require authentication. If you select this option, an additional field appears where you must enter the OAuth token.
- **Username and password**: This option is relevant only if the ESP server is configured to require authentication using a user name and password (SASLogon Services). If you select this option, additional fields appear where you must enter or confirm the user name and password.

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

h If required, select the **Enable server logging** check box to enable logging on the ESP server.

i If required, in the **Number of messages to retain** field, change the default number of messages that are retained by the ESP server log. The default is 10,000 messages.

j Click **OK**.

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 `sourceWindow_01`.

b In the **Description** field, enter `This Source window receives stock trades. This window contains a file/socket connector which reads in the stock trades from a file in CSV format and then publishes the trades to the ESP Engine.`

7 Configure the sourceWindow_01 window's state and event type:

a In the right pane, expand **State and Event Type**.

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

c Select the **Accept only "Insert" events** check box.

If a Source window precedes a Pattern window, you must specify that the Source window is insert-only. This causes the Source window to reject any events with an opcode other than Insert, and permits an index type of pi_EMPTY to be used.

8 Specify an output schema for the sourceWindow_01 window:

a On the right toolbar, 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 the add the next row.

Enter the following values:

Key	Field Name	Type
Y	ID	Int32
N	symbol	String
N	currency	Int32
N	udate	Int64
N	msecs	Int32
N	price	Double
N	quant	Int32
N	venue	Int32
N	broker	Int32
N	buyer	Int32
N	seller	Int32

Key	Field Name	Type
N	buysellflg	Int32
N	trade_time	Timestamp

d Click **OK**.

9 The sourceWindow_01 window will stream a list of vehicles from a file called 50k.csv that contains example data. To add a connector to this CSV file:

a Click  .

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

The Connector Configuration window is displayed.

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

d In the **Fsname** field, enter the full path to the CSV file.

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

f Configure the pub 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 Click **OK**.

g Click **OK**.

10 Expand **Utilities** on the **Windows** pane on the left and drag a Pattern window to the workspace.

The right pane displays the Pattern window's properties.

Pattern windows are insert-only with respect to both their input windows and the output that they produce. As the input and output of a Pattern window are unbounded and insert-only, they are typically stateless windows (that is, windows with index type pi_EMPTY).

11 Specify a name and description for the Pattern window:

a In the right pane, in the **Name** field, change the default name to `patternWindow_01`.

b In the **Description** field, enter `The Pattern window generates pattern matches. The pattern element defines the pattern of interest`.

12 Click **OK**.

13 Connect the sourceWindow_01 window to the patternWindow_01 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 color changes to white.

b Click the white anchor point, hold the mouse button down, and draw a line to the anchor point in the patternWindow_01 window.

The patternWindow_01 window now accepts values from the sourceWindow_01 window.

14 Configure the patternWindow_01 window's pattern of interest:

a Select the patternWindow_01 window on the workspace.

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

c Click  .

A window is displayed enabling you to define the pattern's properties.

i In the **Attributes** section, in the **Name** field, replace the default value with `pattern1`.

ii Select the **Specify a timefield** check box.

iii Confirm that **sourceWindow_01** is selected from the **Source** drop-down list.

iv Confirm that **trade_time** is selected by default from the **Field** drop-down list.

v In the **Events** section, click  .

A panel is displayed that enables you to define an event.

vi In the **Name** field, enter `e1`.

vii In the e1 event's text box, enter `symbol=="GMTC" and s==symbol and p0==price and q0==quant`

Note: Alternatively, you can define event operators by double-clicking the relevant operator in the **Operators** section on the left toolbar.

viii In the **Events** section, click  .

A panel is displayed that enables you to define an event.

ix In the **Name** field, enter `e2`.

x In the e2 event's text box, enter `s==symbol and p0<price*1.5 and q0<quant*1.5 and p1==price and q1==quant`

xi In the **Events** section, click  .

A panel is displayed that enables you to define an event.

xii In the **Name** field, enter `e3`.

xiii In the e3 event's text box, enter `s==symbol and p1<price*1.5 and q1<quant*1.5`

xiv In the **Logic** section, click the panel.

xv Enter `fby{200 milliseconds}(e1, e2, e3)` in the text box.

Note: Alternatively, you can define event logic by double-clicking the relevant event name, operator, or template name on the left toolbar.

xvi On the breadcrumb trail on the toolbar, click **PatternWindow_01**.

The workspace is displayed.

d Specify an output schema for the patternWindow_01 window:

i On the right toolbar, click  .

ii Click  .

The Output Schema and Pattern Mappings window appears.

iii In the **Key Field** section, enter `ID` in the **Name** field.

iv In the **Mapped Fields** section, click  to add a row to the schema table. After you add a row, click  again to the add the next row.

Enter the following values:

Field	Type	Output Type	Event	Value
ID1	Int32	Field-Selection	e1	ID
ID2	Int32	Field-Selection	e2	ID
ID3	Int32	Field-Selection	e3	ID

v Click **OK**.

e Create a subscribe connector:

i Click  .

ii In the right pane, expand **Subscriber Connectors** and click .

The Connector Configuration window is displayed.

iii In the **Name** field, replace the default value with `sub`.

iv Select the **Snapshot** check box.

v In the **Fsname** field, enter the path to the CSV file. For example, you might enter `/opt/sas/viya/home/SASEventStreamProcessingEngine/<release>/examples/xml/pattern_empty_index/output.csv`. Replace *<release>* with the release number in your installation directory path.

vi In the **Fstype** field, enter `csv`.

vii Click **OK**.

15 Configure the project's continuous query:

a Click  .

b Expand **Debugging**.

c In the **Trace in server log** field, select **patternWindow_01**.

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

17 Click ✓ Enter Test Mode.

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

18 In the **ESP Server** drop-down list, select the ESP server on which you want to the model.

19 Click ▷ Run Test.

The results for each window appear on separate tabs:

- the **sourceWindow_01** tab lists the stock trades that are received from the input file.
- the **patternWindow_01** tab lists the matched patterns.

Note: If the table is empty, check that the publisher connector for the sourceWindow_01 window is set correctly to point to the CSV file.

20 To stop the test, click ■ Stop.

The project stops and then unloads from the ESP server.

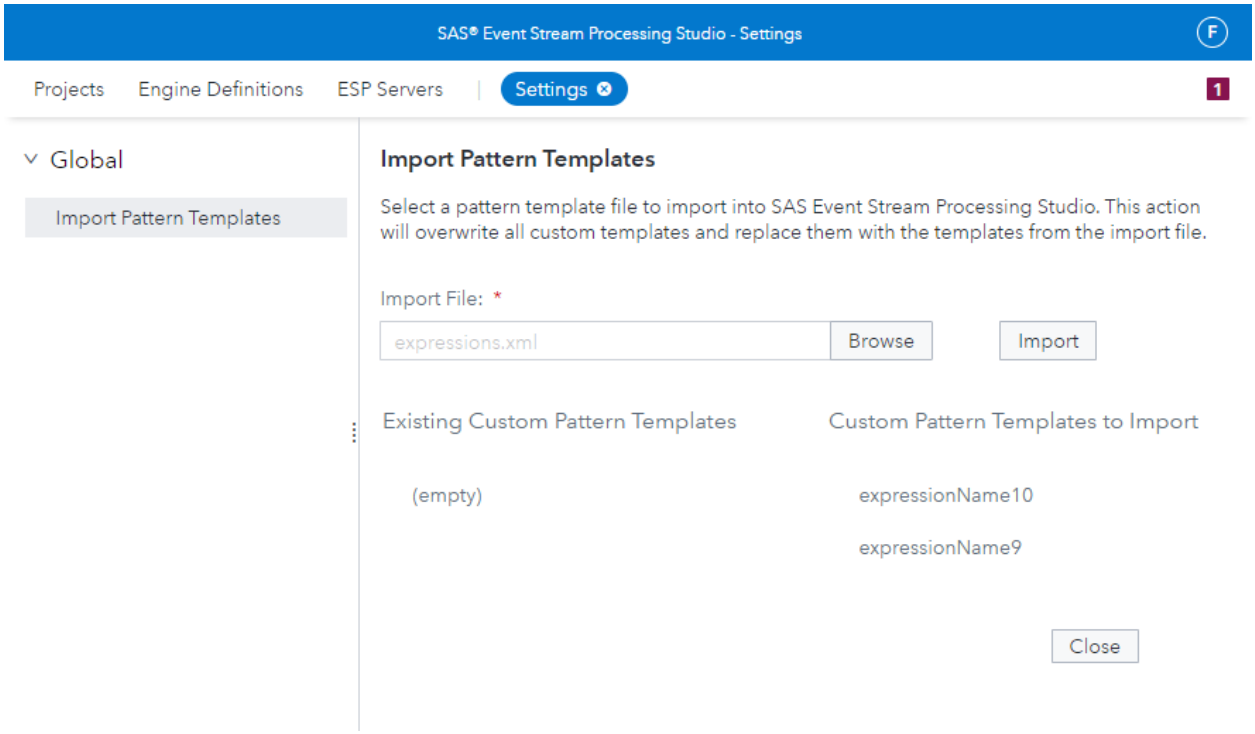
Importing Pattern Templates into SAS Event Stream Processing Studio

You can import custom pattern templates into SAS Event Stream Processing Studio. Pattern templates specify a text string that represents the logical operator expression to combine events of interest (EOIs). The enclosed text specifies an expression that defines the pattern.

The pattern templates that you want to import must be located in an import file. Importing a set of custom pattern templates overwrites and replaces any existing custom pattern templates that were previously imported.

The figure below shows an example of a pattern templates file that contains two custom pattern templates:

Figure 30 *Importing Pattern Templates*



To import a pattern template file:

1 Open the following URL:

`https://host/SASEventStreamProcessingStudio/#/settings`

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

A new tab appears, showing a Settings page.

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

3 Navigate to the file that contains the pattern templates that you want to import and click **Open**.

The **Import Pattern Templates** settings page refreshes to show a list of the custom pattern template present in the file. It also shows any existing custom pattern templates that you previously imported.

4 Position the cursor over a custom pattern template that you want to import.

Notice that the pattern template's logical operator expression appears in a message box.

5 Click **Import**.

A warning message appears, informing you that this action overwrites all existing custom pattern templates and replace them with pattern templates from the file.

6 Click **Yes** to continue.

A message appears, informing you if your import has been successful. The pattern templates that you imported are now available to use when defining a Pattern window's event logic.

Example: Transitioning a Model from Stateful to Stateless

Prepare the Example Files for Use

1. If you have not already done so, download the SAS Event Stream Processing examples package from https://support.sas.com/downloads/package.htm?pid=2421 and extract its contents on your computer.
2. In the examples package, navigate to the `/xml/removeState` directory.
3. Save the InputRemove.csv file on the server on which SAS Event Stream Processing is running. If you want to run the model without following the steps to create it, upload the RemoveState.xml file to SAS Event Stream Processing Studio. For more information, see “Upload a Project” on page 11.

   For more information about these files, see “Transition a Model from Stateful to Stateless Overview” on page 110.

Transition a Model from Stateful to Stateless Overview

This example demonstrates how to facilitate the transition of a stateful part of a model to a stateless part of a model. A Remove State window converts all events that it receives into Inserts and adds a field named eventNumber, which is a monotone-increasing sequential integer. This added field is the only key of the Remove State window.

This example uses two files listed below:

- The XML file (removeState.xml) associated with this example.
- InputRemove.csv is an input file containing the event stream.

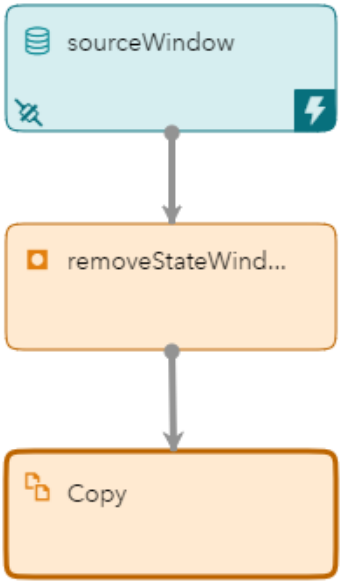
Project Details

This project contains three windows:

- SourceWindow is a Source window. This is where the events from the InputRemove.csv file enter the model.
- removeStateWindow is a Remove State Window. This is where the events are filtered.

- Copy window is a Copy window. This show the transition to a stateless model.

Figure 31 *Diagram of the Transition a Model from Stateful to a Stateless Model.*



Example Steps

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 `RemoveState`.

b Click **OK**.

If you do not currently have any ESP servers configured, you are prompted to decide whether you want to configure an ESP server now. If you are using a Kubernetes cluster, an ESP server is automatically created for you when you run the project in test mode. Therefore you do not need to manually create an ESP server.

Note: It is assumed that you do not have any ESP servers configured. If you already have ESP servers configured or you are going to run the project on an ESP server in a cluster, go to step 5.

3 Click **Yes** to configure an ESP server now or click **No** to skip the ESP server creation process.

If you selected **Yes**, the ESP Server Properties window appears.

4 If you selected **No**, skip this step. If you selected **Yes** to configure an ESP server now, do the following:

a In the **Name** field, enter or update the name that identifies the ESP server.

b In the **Host** field, enter or update the ESP server's host name or IP address.

c In the **HTTP port** field, enter or update the ESP server's administration port number.

d If required, in the **Description** field, enter or update the description of the ESP server.

e If required, in the **Tags** field, enter or update any keywords that describe the ESP server and then press Enter.

f If required, click **Edit** to change the setting for the **Authentication** field:

The Authentication window appears.

- **None**: This is the default option.
- **Kerberos**: This option is relevant only if the ESP server is configured to require authentication using Kerberos.
- **OAuth token**: This option is relevant only if the ESP server is configured to require authentication. If you select this option, an additional field appears where you must enter the OAuth token.
- **Username and password**: This option is relevant only if the ESP server is configured to require authentication using a user name and password (SASLogon Services). If you select this option, additional fields appear where you must enter or confirm the user name and password.

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

h If required, select the **Enable server logging** check box to enable logging on the ESP server.

i If required, in the **Number of messages to retain** field, change the default number of messages that are retained by the ESP server log. The default is 10,000 messages.

j Click **OK**.

5 In the right pane, configure your project's properties: In the **Name** field, change the default name to `Removewindow_proj`.

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 Specify a name for the SourceWindow window: In the right pane, in the **Name** field, change the default name to `SourceWindow`.

8 Specify an output schema for the SourceWindow window:

a On the right toolbar, 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 the add the next row.

Enter the following values:

Key	Field Name	Type
Y	ID	Int32
N	symbol	String
N	true_test	Int32
N	price	Double

d Click **OK**.

9 Configure the SourceWindow window to stream events from a file called InputRemove.csv that contains data. To add a connector to this CSV file:

a In the right pane, click .

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

c Click .

The Connector Configuration window appears.

d In the **Name** field, replace the default value with `pub1`.

e In the **Fsname** field, enter the full path to the CSV file.

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

g Configure the pub1 connector's properties:

i Click **All properties**.

The All Properties window appears.

ii Enter `false` 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 Expand **Transformations** on the **Windows** pane on the left and drag a Remove State window to the workspace.

The right pane displays the Remove State window's properties.

11 Specify a name for the Remove State window. In the right pane, in the **Name** field, change the default name to `removeStateWindow`.

12 Connect the sourceWindow window to the removeStateWindow window with an edge:

a Position the cursor over the anchor point at the bottom of the sourceWindow window so that the anchor point color changes to white.

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

The removeStateWindow window now accepts values from the sourceWindow window.

13 Click the removeStateWindow on the workspace.

14 Configure the removeStateWindow's settings:

a In the right pane, expand **Settings** if it is not already expanded.

b Clear the **Update block deletes** check box.

c Confirm that the **Deletes**, **Retention updates**, and **Retention deletes** check boxes are selected. This setting filters events to pass through the window by their type.

d In addition to these selections, select the **Updates** check box.

e In the **Log Fields** field, select the **Include opcode and flag fields** check box.

15 View the removeStateWindow's output schema:

a On the right toolbar, click  .

b Examine the output schema and notice that the event number, original flag, and original opcode have all been added to the schema. These are strings that specify the opcode and the flags of the event. These two fields directly follow the eventNumber field.

c Click  .

16 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.

17 Specify a name for the Copy window: In the right pane, in the **Name** field, change the default name to Copy.

18 Connect the removeStateWindow window to the Copy window with an edge:

a Position the cursor over the anchor point at the bottom of the removeStateWindow window so that the anchor point color changes to white.

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

The Copy window now accepts values from the removeStateWindow window.

19 Click the Copy on the workspace.

20 Configure the Copy window's state and index:

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

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

21 Configure the Copy window's retention policy:

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

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

c In the **Time limit** field, enter `30` and select **Seconds** from the drop-down list.

22 Configure the project's continuous query:

a Click .

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

c Select the **Instantiate Source windows at run time** check box.

d Expand **Debugging**.

e In the **Enable trace server logging for this query** field, select **removeStateWindow** from the drop-down list.

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

24 Click ✓ Enter Test Mode.

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

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

26 Click ▷ Run Test.

The results for each window appear on separate tabs:

- The **sourceWindow** tab lists all events from the event stream.
- The **removeStateWindow** tab lists the events from the event stream excluding the Delete events that you specified must be removed.
- The **Copy** tab lists retained events.

27 To stop the test, click ■ Stop.

The project stops and then unloads from the ESP server.

Example: Using Text Sentiment Analysis

Prepare the Example Files for Use

1 If you have not already done so, download the SAS Event Stream Processing examples package from https://support.sas.com/downloads/package.htm?pid=2421 and extract its contents on your computer.

2 In the examples package, navigate to the `/xml/text_sentiment` directory.

3 Download the text_sentiment_xml file from `ftp://ftp.sas.com/techsup/download/esp/text_sentiment_xml.zip`.

4 Navigate to the text_sentiment_xml.zip file that you downloaded and extract the sentiment.sam file. Note the location that you extracted the file to.

5 Save the input.csv and sentiment.sam files on the server on which SAS Event Stream Processing is running. If you want to run the model without following the steps to create it, upload the model.xml file to SAS Event Stream Processing Studio. For more information, see “Upload a Project” on page 11.

   For more information about these files, see “Overview” on page 116.

Overview

This model uses Text Sentiment analysis in SAS Event Stream Processing Studio. Text sentiment windows determine the sentiment of text in the specified incoming text field and the probability of its occurrence. The sentiment value is “positive”, “neutral”, or “negative”. The probability is a value between 0 and 1. Text sentiment windows are insert-only.

Note: Without SAS Sentiment Analysis Studio, you do not have the SAM file that is required to initialize a Text Sentiment window.

This example uses three files listed below:

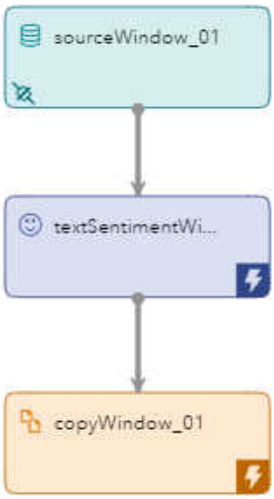
- The XML file (model.xml) associated with this example.
- input.csv is an input file. This file contains the data to be used for sentiment analysis.
- sentiment.sam is a SAM file. This file determines the sentiment of text in the specified incoming text field and the probability of its occurrence.

Project Details

This project contains three windows:

- sourceWindow_01 is a Source window. This is where data from input.csv file enters the model. This data is made available for text sentiment analysis.
- textSentimentWindow is a Text Sentiment window. This is the sentiment of text in the specified incoming text field is determined.
- copyWindow_01 is a Copy window. This is where the textSentiment events are retained according to the specified retention policy.

Figure 32 *Diagram of the Using Text Sentiment Analysis Project*



Example Steps

1 On the **Projects** page, click 

The New Project window appears.

2 In the New Project window, do the following:

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

b Click **OK**.

If you do not currently have any ESP servers configured, you are prompted to decide whether you want to configure an ESP server now. If you are using a Kubernetes cluster, an ESP server is automatically created for you when you run the project in test mode. Therefore you do not need to manually create an ESP server.

Note: It is assumed that you do not have any ESP servers configured. If you already have ESP servers configured or you are going to run the project on an ESP server in a cluster, go to step 5.

3 Click **Yes** to configure an ESP server now or click **No** to skip the ESP server creation process.

If you selected **Yes**, the ESP Server Properties window appears.

4 If you selected **No** skip this step. If you selected **Yes** to configure an ESP server now, do the following:

a In the **Name** field, enter or update the name that identifies the ESP server.

b In the **Host** field, enter or update the ESP server's host name or IP address.

c In the **HTTP port** field, enter or update the ESP server's administration port number.

d If required, in the **Description** field, enter or update the description of the ESP server.

e If required, in the **Tags** field, enter or update any keywords that describe the ESP server and then press Enter.

f If required, click **Edit** to change the setting for the **Authentication** field:

The Authentication window appears.

- **None**: This is the default option.
- **Kerberos**: This option is relevant only if the ESP server is configured to require authentication using Kerberos.
- **OAuth token**: This option is relevant only if the ESP server is configured to require authentication. If you select this option, an additional field appears where you must enter the OAuth token.
- **Username and password**: This option is relevant only if the ESP server is configured to require authentication using a user name and password (SASLogon Services). If you select this option, additional fields appear where you must enter or confirm the user name and password.

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

h If required, select the **Enable server logging** check box to enable logging on the ESP server.

i If required, in the **Number of messages to retain** field, change the default number of messages that are retained by the ESP server log. The default is 10,000 messages.

j Click **OK**.

5 Configure your project's properties:

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

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

c Select **Automatic** from the **Window Publish/subscribe** drop-down list.

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 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 receives a list of events for sentiment analysis.`

8 Click **State and Event Type**.

a In the **Window state and index** field, select **Stateless (pi_EMPTY)**.

b Select the **Accept only "Insert" events** check box.

9 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 the add the next row.

Enter the following values:

Key	Field Name	Type
Y	ID	Int32
N	tstamp	Date
N	msg	String

d Click **OK**.

10 The sourceWindow_01 window will stream a list of vehicles from a file called input.csv that contains example data. To add a connector to this CSV file:

a Click  .

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

The Connector Configuration window is displayed.

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

d In the **Fsname** field, enter the full path to the CSV file.

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

f Configure the pub 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 Click **OK**.

g Click **OK**.

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

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

12 Specify a name and description for the Text Sentiment window:

a In the right pane, in the **Name** field, change the default name to `textSentimentWindow_01`.

b In the **Description** field, enter `This window uses a SAS Text Analytics SAM file to get the sentiment of a document that is in the specified incoming event string field. This window type is insert only. Each input event creates a new text sentiment event.`

13 Connect the sourceWindow_01 window to the textSentimentWindow_01 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 color changes to white.

b Click the white anchor point, hold the mouse button down, and draw a line to the anchor point in the textSentimentWindow_01 window.

The textSentimentWindow_01 window now accepts values from the sourceWindow_01 window.

14 Click the textSentimentWindow_01 window on the workspace.

15 Configure the textSentimentWindow_01 window's sentiment analysis model file full path:

a If it is not already expanded, expand **Text Sentiment**.

b In the **Sentiment analysis model file full path** field, enter the file path to the sentiment analysis model file. For example, enter `/samFiles/sentiment.sam`.

16 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.

17 Specify a name and description for the Copy window:

a In the right pane, in the **Name** field, change the default name to `copyWindow_01`.

b In the **Description** field, enter `This window holds the textSentiment events using a retention policy`.

18 Configure the copyWindow_01 window's state:

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

b Select **Stateful (pi_RBTREE)** from the **Window state and index** drop-down list.

19 Configure the copyWindow_01 window's retention:

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

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

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

20 Connect the textSentimentWindow_01 window to the copyWindow_01 window with an edge:

The copyWindow_01 window now accepts values from the textSentimentWindow_01window.

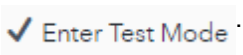
21 Configure the project's continuous query:

a Click  .

b Expand **Debugging**.

c In the **Trace in server log** field, select **textSentimentWindow**.

22 Click  .

23 Click 

.

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

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

25 Click ▷ Run Test.

The results for each window appear on separate tabs:

- the **sourceWindow_01** tab lists the event stream from the input CSV file.
- the **textSentimentWindow** tab lists the textSentiment events.
- the **copyWindow_01** tab lists the textSentiment events according to the specified retention policy.

26 To stop the test, click ■ Stop.

The project stops and then unloads from the ESP server.

Example: Transposing Data from an Aircraft

Prepare the Example Files for Use

1 If you have not already do so, download the SAS Event Stream Processing examples package from https://support.sas.com/downloads/package.htm?pid=2421 and extract its contents on your computer.

2 In the examples package, navigate to the `/xml/transpose_long` and `/xml/transpose_wide` directories.

3 Save the input_long.csv and input_wide.csv files on the server on which SAS Event Stream Processing is running. If you want to run the model without following the steps to create it, upload the uploadTranspose_long.xml and Transpose_wide.xml files to SAS Event Stream Processing Studio. For more information, see “Upload a Project” on page 11.

For more information about these files, see “Transpose Aircraft Information Example Overview” on page 121.

Transpose Aircraft Information Example Overview

This example conceptualizes an event as a row that consists of multiple columns. You can use a Transpose window to interchange an event’s rows as columns, and columns as rows. Use attributes of the Transpose window to govern the rearrangement of data. You will process information about the pitch, yaw, roll, and velocity of an aircraft in flight.

As the Transpose window has two modes, long and wide, this example consists of two parts:

- Transpose aircraft information in wide mode. For more information, see “Wide Mode Example Steps” on page 122.

- Transpose aircraft information in long mode. For more information, see “Long Mode Example Steps” on page 128.

The long mode part of this example uses the following files:

- The XML file (transpose_long.xml) associated with this example.
- input_long.csv. This is an input file. This file contains event streams from the aircraft in flight.

The wide mode part of this example uses the following files:

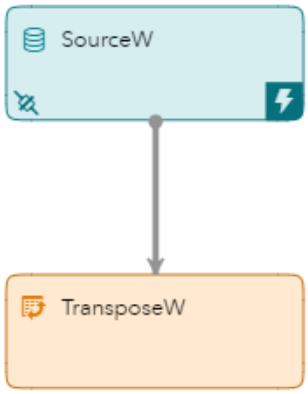
- The XML file (transpose_wide.xml) associated with this example.
- input_wide.csv. This is an input file. This file contains event streams from the aircraft in flight.

Project Details

This project contains two windows:

- A Source window, where aircraft events from the input file enter the model. In the wide mode version of this example, this file is called input_wide.csv. In the long mode version of this example, this file is called input_long.csv.
- A Transpose window, where the transposition of the aircraft events occurs. You can configure the attributes of the Transpose window to govern the rearrangement of data. In the wide mode version of this example, this window is called TransposeW. In the long mode version of this example, this window is called TranposeL.

Figure 33 *Diagram of the Transpose Model in Wide Mode*



Wide Mode Example Steps

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 `Transpose_wide`.

b In the **Description** field, enter a description: `This example transposes information in wide mode about the pitch, yaw, roll, and velocity of an aircraft in flight.`

c Click **OK**.

If you do not currently have any ESP servers configured, you are prompted to decide whether you want to configure an ESP server now.If you are using a Kubernetes cluster, an ESP server is automatically created for you when you run the project in test mode. Therefore you do not need to manually create an ESP server.

Note: It is assumed that you do not have any ESP servers configured. If you already have ESP servers configured or you are going to run the project on an ESP server in a cluster, go to step 5.

3 Click **Yes** to configure an ESP server now or click **No** to skip the ESP server creation process.

If you selected **Yes**, the ESP Server Properties window appears.

4 If you selected **No**, skip this step. If you selected **Yes** to configure an ESP server now, do the following:

a In the **Name** field, enter or update the name that identifies the ESP server.

b In the **Host** field, enter or update the ESP server's host name or IP address.

c In the **HTTP port** field, enter or update the ESP server's administration port number.

d If required, in the **Description** field, enter or update the description of the ESP server.

e If required, in the **Tags** field, enter or update any keywords that describe the ESP server and then press Enter.

f If required, click **Edit** to change the setting for the **Authentication** field:

The Authentication window appears.

- **None**: This is the default option.
- **Kerberos**: This option is relevant only if the ESP server is configured to require authentication using Kerberos.
- **OAuth token**: This option is relevant only if the ESP server is configured to require authentication. If you select this option, an additional field appears where you must enter the OAuth token.
- **Username and password**: This option is relevant only if the ESP server is configured to require authentication using a user name and password (SASLogon Services). If you select this option, additional fields appear where you must enter or confirm the user name and password.

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

h If required, select the **Enable server logging** check box to enable logging on the ESP server.

i If required, in the **Number of messages to retain** field, change the default number of messages that are retained by the ESP server log. The default is 10,000 messages.

j Click **OK**.

5 Configure the project's threading level:

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

b In the **Threads** field, enter `2`.

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.

This window receives information about an aircraft.

7 Specify a name for the Source window: in the right pane, in the **Name** field, change the default name to `SourceW`.

8 Change the SourceW window's state and event type:

a Expand **State and Event Type**.

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

c Select the **Accept only "Insert" events** check box.

9 Specify an output schema for the SourceW 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	PlaneID	String
N	TAG	String
N	value	Double
N	time	Stamp
N	lat	Double
N	long	Double

- The `PlaneID` field of the schema identifies which aircraft provides the data.
- The `TAG` field specifies whether the data contains the aircraft's pitch, yaw, roll, or velocity.
- The `value` field records the numerical value for the `TAG` and the specific time that the data is recorded.

- The event captured the plane's latitude (lat) and longitude (long) when the pitch, yaw, roll, or velocity is recorded.

d Click **OK**.

10 Configure the SourceWindow window to stream events from a file called input.csv that contains data from an aircraft in flight. To add a connector to this CSV file:

a In the right pane, click  .

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

c Click  .

The Connector Configuration window appears.

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

e In the **Fsname** field, enter the full path to the CSV file.

f In the **Fstype** field, enter **csv**.

g Configure the pub connector's properties:

i Click **All properties**.

The All Properties window appears.

ii Select `false` in the **Value** field of the **transactional** property.

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

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

v Enter `1` in the **Value** field of the **Rate** property.

vi Click **OK**.

h Click **OK**.

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

12 Open the input_wide.csv file and examine its contents:

```
i,n,1,turboprop #1, pitch,1.1, 2017-08-30 15:21:00.000000,10,10
i,n,2,turboprop #2, velocity,-1.2, 2017-08-30 15:21:00.000001,20,20
i,n,3,turboprop #1, roll,-1.1, 2017-08-30 15:21:00.000002,11,11
i,n,4,turboprop #2, yaw,2.2, 2017-08-30 15:21:00.000003,21,21
i,n,5,turboprop #2, pitch,1.2, 2017-08-30 15:21:00.000004,22,22
i,n,6,turboprop #1, velocity,-1.1, 2017-08-30 15:21:00.000005,12,12
i,n,7,turboprop #2, roll,-1.2, 2017-08-30 15:21:00.000006,23,23
i,n,8,turboprop #1, yaw,2.1, 2017-08-30 15:21:00.000007,13,13
i,n,9,jet #1, pitch,1.3, 2017-08-30 15:21:00.000012,30,30
i,n,10,jet #2, velocity,-4.4, 2017-08-30 15:21:00.000013,40,40
i,n,11,jet #1, roll,-4.3, 2017-08-30 15:21:00.000014,31,31
i,n,12,jet #2, yaw,8.4, 2017-08-30 15:21:00.000015,41,41
i,n,13,jet #2, pitch,4.4, 2017-08-30 15:21:00.000016,42,42
i,n,14,jet #1, velocity,-4.3, 2017-08-30 15:21:00.000017,32,32
i,n,15,jet #2, roll,-4.4, 2017-08-30 15:21:00.000018,43,43
i,n,16,jet #1, yaw,8.3, 2017-08-30 15:21:00.000019,33,33
i,n,17,turboprop #1, pitch,23.1, 2017-08-30 15:21:06.000022,14,14
```

In these seventeen events, four planes stream data through the Source window: **`turboprop #1`**, **`turboprop #2`**, **`jet #1`**, and **`jet #2`**. Each event contains either a pitch, velocity, roll, or yaw value at a specific time.

- `Wide` mode produces output events that contain multiple columns, each based on data streamed through the input events.
- The values of `TAG` in the input events determine columns in output events.
- `Tags-included` specify which specific values of `TAG` are to be included in the output events. Here, all four values of `TAG` are specified.
- The `value` and `time` associated with pitch, yaw, roll, and velocity are included in the output event. The associated latitude and longitude are passed through.
- Output events are grouped by the value of `PlaneID`.

Output columns are formed by taking the cross product of the following:

```
{pitch, yaw, roll, velocity} times {value, time}
```

This yields pitch_value, pitch_time, yaw_value, yaw_time, and so on.

13 Expand **Transformations** on the **Windows** pane on the left and drag a Transpose window to the workspace.

The right pane displays the Transpose window's properties.

The input data is to be streamed through a Transpose window to create a single event (row). This event contains the pitch, velocity, roll, and yaw of each aircraft at a specific time.

14 Specify a name and description for the Transpose window. In the right pane, in the **Name** field, change the default name to **`TransposeW`**.

15 Connect the SourceW window to the TransposeW window with an edge:

a Position the cursor over the anchor point at the bottom of the SourceW window so that the anchor point color changes to white.

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

The TransposeW window now accepts values from the SourceW window.

16 Click the TransposeW window on the workspace.

17 Configure the TransposeW window's settings:

a If necessary, click **Settings** to expand the section.

b Confirm the **Mode** field is set to **Wide**.

c In the **Tag name** field, select **TAG** from the drop-down list.

d In the **Included tags** field, enter the following tag names: **`pitch`**, **`roll`**, **`yaw`**, and **`velocity`**. After you have entered each tag name, press Enter to confirm its creation.

Note: Ensure you enter each tag in all lowercase characters.

e In the **Tag values** field, double-click the row in the table

The Select Tag Value Fields window appears.

i Select the **value** and **time** fields.

ii Click **OK** to confirm your selection.

f In the **Group By** field, select **PlaneID** from the drop-down list.

18 Configure a subscriber connector for the TransposeW window:

a Expand **Subscriber Connectors**.

b Click .

The Connector Configuration window appears.

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

d In the **Fsname** field, enter the full path to the output file: **result.out**.

e In the **Fstype** field, enter **csv**.

f Configure the sub 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 Click **OK**.

g Click **OK**.

19 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 `transpose_cq`.

c Expand **Debugging**.

d In the **Trace in server log** field, select **TransposeW** from the drop-down list.

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

21 Click ✓ Enter Test Mode.

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

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

23 Click ▷ Run Test.

The results for each window appear on separate tabs:

- The **SourceW** tab shows events representing the data received from the aircraft.
- The **TransposeW** tab shows the transposed aircraft data in wide format.

24 To stop the test, click ■ Stop.

The project stops and is unloaded from the ESP server.

Long Mode Example Steps

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 `Transpose_long`.

b In the **Description** field, enter a description: `This example transposes information in long mode about the pitch, yaw, roll, and velocity of an aircraft in flight.`

c Click **OK**.

If you do not currently have any ESP servers configured, you are prompted to decide whether you want to configure an ESP server now.If you are using a Kubernetes cluster, an ESP server is automatically created for you when you run the project in test mode. Therefore you do not need to manually create an ESP server.

Note: It is assumed that you do not have any ESP servers configured. If you already have ESP servers configured or you are going to run the project on an ESP server in a cluster, go to step 5.

3 Click **Yes** to configure an ESP server now or click **No** to skip the ESP server creation process.

If you selected **Yes**, the ESP Server Properties window appears.

4 If you selected **No**, skip this step. If you selected **Yes** to configure an ESP server now, do the following:

a In the **Name** field, enter or update the name that identifies the ESP server.

b In the **Host** field, enter or update the ESP server's host name or IP address.

c In the **HTTP port** field, enter or update the ESP server's administration port number.

d If required, in the **Description** field, enter or update the description of the ESP server.

e If required, in the **Tags** field, enter or update any keywords that describe the ESP server and then press Enter.

f If required, click **Edit** to change the setting for the **Authentication** field:

The Authentication window appears.

- **None**: This is the default option.
- **Kerberos**: This option is relevant only if the ESP server is configured to require authentication using Kerberos.
- **OAuth token**: This option is relevant only if the ESP server is configured to require authentication. If you select this option, an additional field appears where you must enter the OAuth token.
- **Username and password**: This option is relevant only if the ESP server is configured to require authentication using a user name and password (SASLogon Services). If you select

this option, additional fields appear where you must enter or confirm the user name and password.

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

h If required, select the **Enable server logging** check box to enable logging on the ESP server.

i If required, in the **Number of messages to retain** field, change the default number of messages that are retained by the ESP server log. The default is 10,000 messages.

j Click **OK**.

5 Configure your project's threading level:

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

b In the **Threads** field enter `2`.

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.

This window receives events about

7 Specify a name for the Source window: in the right pane, in the **Name** field, change the default name to `SourceW`.

8 Change the SourceW window's state and event type:

a Expand **State and Event Type**.

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

c Select the **Accept only "Insert" events** check box.

9 Specify an output schema for the SourceW 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	pitch_value	Double
N	pitch_time	Stamp
N	yaw_value	Double

Key	Field Name	Type
N	yaw_time	Stamp
N	roll_value	Double
N	roll_time	Stamp
N	velocity_value	Double
N	velocity_time	Stamp
N	lat	Double
N	long	Double

- The event captures the value of the aircraft's pitch, yaw, roll, and velocity along with the time at which they were recorded.
- The event captures the plane's latitude (lat) and longitude (long) when the pitch, yaw, roll, or velocity is recorded.

d Click **OK**.

10 Configure the SourceWindow window to stream events from a file called input_long.csv that contains data from an aircraft in flight. To add a connector to this CSV file:

a In the right pane, click  .

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

c Click  .

The Connector Configuration window appears.

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

e In the **Fsname** field, enter the full path to the CSV file.

f In the **Fstype** field, enter **csv**.

g Configure the pub connector's properties:

i Click **All properties**.

The All Properties window appears.

ii Select `false` in the **Value** field of the **transactional** property.

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

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

v Click **OK**.

h Click **OK**.

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

12 Open the input_long.csv file and examine its contents:

```
I,N,1,23.100000,2017-08-30 15:21:06.000022,2.100000,2017-08-30 15:21:00.000007,
-1.100000,2017-08-30 15:21:00.000002,-1.100000,2017-08-30
15:21:00.000005,10.000000,10.000000
```

- `Long` mode produces one or more event per incoming event.
- The `value` and `time` associated with pitch, yaw, roll, and velocity are included in the output event. The associated latitude and longitude are passed through.
- Output events are grouped by the value of `ID`.

When you use long mode, you obtain the inverse results of wide mode. The Transpose window streams a number of events for each wide event that it receives. Input schema for the Source window must reflect combinations of fields.

13 Expand **Transformations** on the **Windows** pane on the left and drag a Transpose window to the workspace.

The right pane displays the Transpose window's properties.

The input data is to be streamed through a Transpose window to create a single event (row). This event contains the pitch, velocity, roll, and yaw of each aircraft at a specific time.

14 Specify a name and description for the Transpose window. In the right pane, in the **Name** field, change the default name to **`TransposeL`**.

15 Connect the SourceW window to the TransposeL window with an edge:

a Position the cursor over the anchor point at the bottom of the SourceW window so that the anchor point color changes to white.

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

The TransposeL window now accepts values from the SourceW window.

16 Click the TransposeL window on the workspace.

17 Configure the TransposeL window's settings:

a If necessary, click **Settings** to expand the section.

b In the **Mode** field, select **Long**.

c In the **Tag name** field, enter **`TAG`**.

d In the **Included values** field, enter the following tag names: **value** and **time**. After you have entered each tag name, press Enter to confirm its creation.

e In the **Included tags** field, enter the following tag names: **pitch**, **roll**, **yaw**, and **velocity**. After you have entered each tag name, press Enter to confirm its creation.

f Collapse **Settings**.

18 Create a subscribe connector:

a In the right pane, expand **Subscriber Connectors** and click .

The Connector Configuration window is displayed.

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

c In the **Fsname** field, enter the full path to the CSV file.

d In the **Fstype** field, enter **csv**.

e Configure the sub 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 **Rate** property.

iv Click **OK**.

f Click **OK**.

19 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 `transpose_cq`.

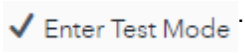
20 Configure your project's debugging properties:

a Click  .

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

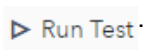
c In the **Enable trace server logging for this query** field, select **tranposeL** from the drop-down list.

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

22 Click  .

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

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

24 Click  .

The results for each window appear on separate tabs:

- The **SourceW** tab shows events representing the data received from the aircraft.
- The **TransposeL** tab shows the transposed aircraft data in long format.

25 To stop the test, click  .

The project stops and is unloaded from the ESP server.

Example: Performing Multi-Object Tracking (MOT)

Prepare the Example Files for Use

1. If you have not already done so, download the SAS Event Stream Processing examples package from https://support.sas.com/downloads/package.htm?pid=2421 and extract its contents on your computer.
2. In the examples package, navigate to the `/xml/object_tracker` directory.
3. Save the det.txt and MOT17-04-FRCNN_1.csv files on the server on which SAS Event Stream Processing is running. If you want to run the model without following the steps to create it, upload the model.xml file to SAS Event Stream Processing Studio. For more information, see “Upload a Project” on page 11.

   For more information about these files, see “Overview” on page 133.

Overview

This example uses an Object Tracking window to perform multi-object tracking (MOT). MOT is the process of accurately estimating the identity and position of multiple objects over time using a model that is based on incoming observations. The Object Tracking window couples trackers with detectors in a process called tracking-by-detection. Specifically, it uses an intersection-over-union (IOU) method of tracking-by-detection. In the tracking-by-detection method, an object detector is applied to each frame in a video. A tracker is then used to assign these detections to tracks. The IOU method tracks objects without using image information. Instead, it relies only on detection data. Using this method enables incremental and event-oriented tracking.

This example uses the following seven files:

- The XML file (model.xml) associated with this example.
- det.txt is an input file. This file contains detection data.
- addObjId.csv is an output file. When you run the model, the object IDs are written to the addObjid.csv file.
- MOT17–04–FRCNN_1.csv is an input file. This file is the result of converting a video file into a CSV file. Each line in the CSV file represents a frame in the video.
- outputTracks.csv is an output file. When you run the model, the output tracks are written to the outputTracks.csv file.
- transpLong.csv is an output file. When you run the model, transposed output tracks are written to the transpLong.csv file.

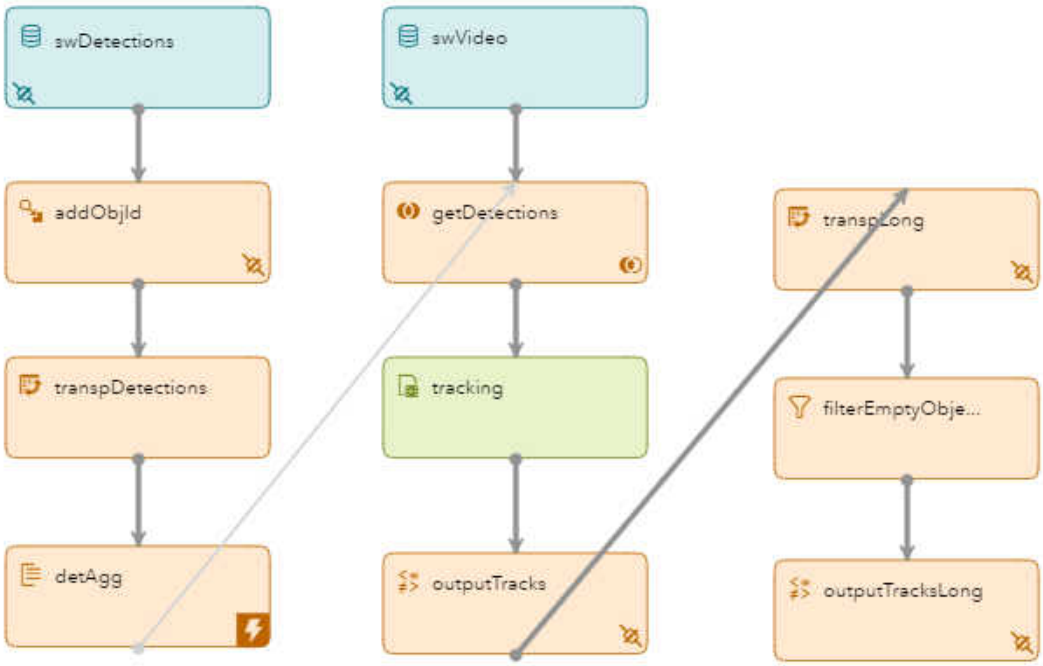
- outputTracksLong.csv is an output file. When you run the model, output tracks in long format are written to the outputTracksLong.csv file.

Project Details

The project contains eleven windows:

- swDetections is a Source window. This is where detection data from the det.txt file enters the model.
- addObjId is a Compute window. This is where an object ID is assigned to each detection in the video file. A list of object IDs is output to the addObjId.csv file.
- transpDetections is a Transpose window. This is where the detection data is transposed using each detection's object ID.
- detAgg is an Aggregate window. This is where the detection data is aggregated.
- swVideo is a Source window. This is where data from the MOT17–04–FRCNN_1.csv file enters the model. This file is the result of converting a video file into a CSV file. Each line in the file represents one frame in the video.
- getDetections is a Join window. This is where the video frames from the MOT17–04–FRCNN_1.csv file and the detection data from the det.txt file are merged.
- tracking is an Object Tracker window. This is where incoming detection data is used to perform multi-object tracking. This window uses an input schema to propose detections.
- outputTracks is a Functional window. This is where the output tracks are written. A list of output tracks are output to the outputTracks.csv file.
- transpLong is Transpose window. This is where the output tracks are transposed and written to the transpLong.csv file.
- filterEmptyObjects is a Filter window. This is where objects that do not contain any values are filtered out of the model.
- outputTracksLong is a Functional window. This is where output tracks are written in long format to the outputTracksLong.csv file.

Figure 34 *Diagram of the Perform Multi-object Tracking (MOT) in Real-Time Model*



Example Steps

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 `MOT17_tracking`.

b Click **OK**.

If your deployment does not contain any configured ESP servers, you are prompted to decide whether you want to configure an ESP server now. If you are using a Kubernetes cluster, an ESP server is automatically created for you when you run the project in test mode. Therefore, you do not need to manually create an ESP server.

Note: It is assumed that your deployment does not contain any configured ESP servers. If your deployment already contains a configured ESP server or you are going to run the project on an ESP server in a cluster, go to step 5.

3 Click **Yes** to configure an ESP server now or click **No** to skip the ESP server creation process.

If you selected **Yes**, the ESP Server Properties window appears.

4 If you selected **No**, skip this step. If you selected **Yes** to configure an ESP server now, do the following:

a In the **Name** field, enter or update the name that identifies the ESP server.

b In the **Host** field, enter or update the ESP server's host name or IP address.

c In the **HTTP port** field, enter or update the ESP server's administration port number.

d If required, in the **Description** field, enter or update the description of the ESP server.

e If required, in the **Tags** field, enter or update any keywords that describe the ESP server and then press Enter.

f If required, click **Edit** to change the setting for the **Authentication** field:

The Authentication window appears.

- **None**: This is the default option.
- **Kerberos**: This option is relevant only if the ESP server is configured to require authentication using Kerberos.
- **OAuth token**: This option is relevant only if the ESP server is configured to require authentication. If you select this option, an additional field appears where you must enter the OAuth token.
- **Username and password**: This option is relevant only if the ESP server is configured to require authentication using a user name and password (SASLogon Services). If you select this option, additional fields appear where you must enter or confirm the user name and password.

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

h If required, select the **Enable server logging** check box to enable logging on the ESP server.

i If required, in the **Number of messages to retain** field, change the default number of messages that are retained by the ESP server log. The default is 10,000 messages.

j Click **OK**.

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

a Expand **Attributes**.

b In the **Default window state and index** field, select **Stateless (pi_EMPTY)**.

c Collapse **Attributes**.

d Expand **User-Defined Properties**.

e Click  to add a row to the table.

f Click the cell in the Name column.

g Remove the default text and enter `DATA_DIR`.

h Click the cell in the Value column.

i Enter an appropriate location on your file system where the example's data files can be stored.

6 Configure the project's continuous query:

a Click  .

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

c Select the **Instantiate unconnected Source windows at run time** check box.

Selecting this option instantiates singleton Source windows at run time. A Source window that is defined within a continuous query but not included in an edge element is called a singleton. Singletons can be useful to validate a connection between the outside world and the ESP server and validating input formats when designing a project.

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.

8 Specify a name for the Source window:

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

b Collapse **Name and Description**.

9 Configure the swDetections window's state and event type:

a Expand **State and Event Type**.

b In the **Window state and index** field, select **Stateless (pi_EMPTY)**.

c Select the **Accept only "Insert" events** check box.

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

e Collapse **State and Event Type**.

10 Configure the swDetections window to stream events from a file called det.txt that contains detection data. To add a connector to this file:

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

b Click  .

The Connector Configuration window appears.

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

d In the **Fsname** field, enter the full path to the file. Alternatively, you might enter `@DATA_DIR@/det.txt`.

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

f Click **All properties**:

The All Properties window appears.

i In the **noautogenfield** property's **Value** field, select **true** from the drop-down list.

ii In the **addcsvopcode** property's **Value** field, select **true** from the drop-down list.

iii In the **addcsvflags** property's **Value** field, select **normal** from the drop-down list.

iv Click **OK**.

g Click **OK**.

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

11 Specify an output schema for the swDetections 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:

Table 2 A Table Displaying Output Schema Values for the SwDetections Window

Key	Field Name	Type
Yes	id	Int64
No	frame	Int64
No	a	Int32
No	x	Double
No	y	Double
No	w	Double
No	h	Double
No	score	Double

d Click **OK**.

e Click .

12 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.

13 Specify a name for the Compute window:

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

b Collapse **Name and Description**.

14 Connect the swDetections window to the addObjId window with an edge:

a Position the cursor over the anchor point at the bottom of the swDetections window so that the anchor point color changes to white.

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

The addObjId window now accepts values from the swDetections window.

15 Click the addObjId window on the workspace.

16 Configure the addObjId window's state:

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

b From the **State** field, select **Stateless (pi_EMPTY)** from the drop-down list.

c Collapse **State**.

17 Configure the addObjId window's compute settings:

a Expand **Compute Settings**.

b In the **Compute method** field, confirm that **Expressions** is selected from the drop-down list.

c Select the **Include engine initialization expression** check box.

d In the **Return type** field, confirm that **string** is selected.

e In the **Expression** field, enter:

```
integer last_frame; last_frame = 0;
            integer obj_number; obj_number = 0;
```

In this expression, you have declared the variables last_frame and obj_number and assigned these variables the value of 0.

f Click  to validate the expression.

A message appears, informing you of the expression's validity.

Note: To validate your expression successfully, a working ESP server must be available. If you are using only a Kubernetes environment, skip this step as expression validation is not supported for a project running on a cluster.

g Confirm that the expression is valid.

h Select the **Include user-defined functions** check box.

i Click  .

The User-Defined Function Properties window appears.

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

k Click .

The Expression Editor appears.

l Enter the following in the text box:

```
private integer curr_frame;
              curr_frame = parameterstring(1);
              if curr_frame !=last_frame then begin;
                obj_number = 0;
              end;
              else begin;
                obj_number = obj_number + 1;
              end;
              last_frame = curr_frame;
              return obj_number;
```

In this expression, you have defined a function to return the object number from the current frame in the video file.

m Click 

A message appears, informing you of the expression's validity.

Note: To validate your expression successfully, a working ESP server must be available. If you are using only a Kubernetes environment, skip this step as expression validation is not supported for a project running on a cluster.

n Confirm that the expression is valid.

o Click **OK**.

p In the **Return type** field, select **int32** from the drop-down list.

q Click **OK**.

r Collapse **Compute Settings**.

18 Configure a subscriber connector for the addObjId window.

a Expand **Subscriber Connectors**.

b Click  .

The Connector Configuration window appears.

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

d In the **Fsname** field, enter `@DATA_DIR@/addObjId.csv`.

e In the **Fstype** field, enter `csv`.

f Click **All Properties**:

i In the **header** property's **Value** field, select **full** from the drop-down list.

ii Click **OK**.

g Click **OK**.

19 Specify an output schema for the addObjId window:

a In the right pane, click  .

b Click  .

The Output Schema window appears.

c Click  .

The Copy Fields From Input Schema window appears.

d Select the following fields:

Note: Do not select the **a** field.

Table 3 A Table Displaying Output Schema Values for the addObjId Window

Window	Field	Type
swDetections	id	Int64
swDetections	frame	Int64
swDetections	x	Double
swDetections	y	Double
swDetections	w	Double
swDetections	h	Double
swDetections	score	Double

e Click **OK**.

You are returned to the Output Schema window.

f Locate the **Frame** field in the table and change its default expression to `frame-1`.

g 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:

Table 4 A Table Displaying Output Schema Field Expressions for the addObjId Window

Key	Field Name	Type	Expression
No	objId	string	'obj_' & getObjNb(frame)
No	objCount	int32	obj_number+1
No	label	string	'person'

h Click **OK**.

i Click  .

20 Expand **Transformations** on the **Windows** pane on the left and drag a Transpose window to the workspace.

The right pane displays the Transpose window's properties.

21 Specify a name for the Transpose window:

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

b Collapse **Name and Description**.

22 Connect the addObjId window to the transpDetections window with an edge:

a Position the cursor over the anchor point at the bottom of the addObjId window so that the anchor point color changes to white.

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

The transpDetections window now accepts values from the addObjId window.

23 Click the transpDetections window on the workspace.

24 Configure the transpDetections window's settings:

a Expand **Settings** if it not already expanded.

b In the **Mode** field, confirm that **Wide** is selected.

c In the **Tag name** field, select **objId** from the drop-down list.

d In the **Tag values** field, click the table.

The Select Tag Value Fields window appears.

e Select the following fields:

Table 5 A Table Displaying Tag Values to Be Selected

Field	Type
x	Double
y	Double
w	Double
h	Double
score	Double
label	String

f Click **OK** to close the Select Tag Values Fields window.

g In the **Clear timeout** field, confirm that **Never** is selected.

h In the **Group by** field, enter **frame**.

i Click  .

The XML Editor appears.

j Copy and paste the following XML code into the `<window-transpose>` XML tag:

```
tags-included="obj_0,obj_1,obj_2,obj_3,obj_4,obj_5,obj_6,obj_7,
obj_8,obj_9,obj_10,obj_11,obj_12,obj_13,obj_14,obj_15,obj_16,
obj_17,obj_18,obj_19,obj_20,obj_21,obj_22,obj_23,obj_24,obj_25,
obj_26,obj_27,obj_28,obj_29,obj_30,obj_31,obj_32,obj_33,obj_34,
```

```
obj_35,obj_36,obj_37,obj_38,obj_39,obj_40,obj_41,obj_42,obj_43,
obj_44,obj_45,obj_46,obj_47,obj_48,obj_49,obj_50,obj_51,obj_52,
obj_53,obj_54,obj_55,obj_56,obj_57,obj_58,obj_59,obj_60,obj_61,
obj_62,obj_63,obj_64,obj_65"
```

Copying and pasting the XML code above using the XML Editor is quicker and more efficient that manually entering each tag using the **Included tags** field in the user interface.

Note: Ensure you paste the above XML code before the closing XML tag `/>`.

k Click  .

25 Examine the **Included tags** field and note that the tags that you just created using the XML Editor appear in the field.

26 Inspect the transpDetections output schema:

a In the right pane, click  .

b Examine the output schema and note that the tags and tag values that you just created are now included in the schema.

c Click  .

27 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.

28 Specify a name for the Aggregate window:

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

b Collapse **Name and Description**.

29 Configure the detAgg window's state:

a Expand **State**.

b In the **Window state and index** field, select **Stateful (pi_HASH)**.

30 Examine the detAgg window's aggregation settings

a Click **Aggregation Settings**.

b In the **Aggregation method** field, confirm that **Expressions** is selected.

31 Connect the transpDetections window to the detAgg window with an edge:

a Position the cursor over the anchor point at the bottom of the transpDetections window so that the anchor point color changes to white.

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

The detAgg window now accepts values from the transpDetections window.

32 Click the detAgg window on the workspace.

33 Specify an output schema for the detAgg window:

a In the right pane, click  .

b Click  .

The Output Schema window appears.

c Click  .

The Copy Fields From Input Schema window appears.

d Select all of the fields in the table with the exception of the **id** field.

e Click **OK**.

You are returned to the Output Schema window.

f Select the **frame** field to be the key field.

g Select the **objCount** field.

h Click  to move the **objCount** field to be the second field in the schema.

i Click **OK**.

j Examine the output schema. Each field in the schema now has an aggregation value.

k Click  .

34 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.

35 Specify a name for the Source window:

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

b Collapse **Name and Description**.

36 Configure the swVideo window's state and event type:

a Expand **State and Event Type**.

b In the **Window state and index** field, select **Stateless (pi_EMPTY)**.

c Select the **Accept only "Insert" events** check box.

37 Configure the swVideo window to stream events from a file called MOT17–04–FRCNN_1.csv that contains frames from a video file that has been converted into CSV format. To add a connector to this file:

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

b Click  .

The Connector Configuration window appears.

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

d In the **Fsname** field, enter the full path to the file. Alternatively, you might enter `@DATA_DIR@/MOT17-04-FRCNN_1.csv`.

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

f Click **OK**.

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

38 Specify an output schema for the swVideo 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:

Table 6 A Table Displaying Output Schema Values for the swVideo Window

Key	Field Name	Type
Y	ID	Int64
N	image	Blob

d Click **OK**.

e Click  .

39 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.

40 Specify a name for the Join window: In the right pane, in the **Name** field, change the default name to `getDetections`.

41 Connect the swVideo window to the getDetections window with an edge:

a Position the cursor over the anchor point at the bottom of the swVideo window so that the anchor point color changes to white.

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

The getDetections window now accepts values from the swVideo window.

42 Connect the detAgg window to the getDetections window with an edge:

a Position the cursor over the anchor point at the bottom of the detAgg window so that the anchor point color changes to white.

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

The getDetctions window now accepts values from the detAgg window.

43 Click the getDetections window on the workspace.

44 Examine the getDetections window's settings:

a Expand **Settings**.

b Inspect the **Left window** and **Right window** fields. Notice that the swVideo window is regarded as the left window and the detAgg 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, confirm that **Select fields** is selected from the drop-down list.

As a result of choosing the **Select fields** option, as new input events arrive, the non-key fields for the join are calculated using a join selection string. This selection string is a one-to-one mapping of input fields to join fields.

d Collapse **Settings**.

45 Configure the getDetections window's join criteria:

a Expand **Join Criteria**.

b In the **Join type** field, confirm that **Left Outer** is selected.

c Select the **Set the "no re-generates" option** check box.

The default join behavior is to always regenerate the appropriate rows of a Join window when a change is made to either side of the joins. With no-regeneration mode, when there is a left outer join on a change to the right window (lookup side), changes to the dimension (lookup) table affect only new fact events. All previous fact events that have been processed by the join are not regenerated. This frequently occurs when a new dimension window is periodically flushed and re-loaded. Setting this flag to true for your Join window enables the no-regenerates semantics.

d Collapse **Join Criteria**.

46 Configure the getDetections 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: swVideo column twice, and select **ID** from the drop-down list.

d Click the cell in the Right: detAgg column twice, and select **frame** from the drop-down list.

47 Specify an output schema for the getDetections window:

a In the right pane, click  .

b Click .

The Edit Output Schema window appears.

c Click .

The Copy Fields From Input Schema window appears.

d Select all of the fields in the table with the exception of the **ID** field and the **frame** field.

e Click **OK**.

You are returned to the Edit Output Schema window.

f Click **OK**.

g Click  .

48 Expand **Utilities** on the **Windows** pane on the left and drag an Object Tracker window to the workspace.

The right pane displays the Object Tracker window's properties.

49 Specify a name for the Object Tracker window:

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

b Collapse **Name and Description**.

50 Configure the tracking window's settings:

a Expand **Settings**.

b In the **Mode** field, select **Wide**.

c In the **Output tracks** field, change the default entry to `65`.

In the tracking-by-detection method, an object detector is applied to each frame in a video. A tracker is then used to associate these detections to tracks. In this example, 65 tracks are used.

d Select the **Output velocity vector coordinates** check box.

Selecting this option displays the coordinates of the last velocity vector of the object.

e Collapse **Settings**.

51 Configure the tracking window's tracking properties:

a Expand **Tracking Properties**.

b In the **Tracking Method** field, confirm that **IOU** is displayed.

This example uses the intersection-over-union (IOU) tracking method. Tracking using this method is implemented by associating detections by their spatial overlap between time steps.

c In the **Sigma score** section, in the **Low** field, confirm that **0.3** is the default entry.

The sigma score **Low** field specifies the low detection threshold value (σl value).

d In the **Sigma score** section, in the **High** field, change the default entry to `0.7`.

The sigma score **High** field specifies the high detection threshold value (σh value).

e In the **Sigma thresholds** section, in the **First** field, change the default entry to `0.6`.

The **First** field specifies the first IOU threshold value (σIOU value).

f In the **Sigma thresholds** section, in the **Second** field, confirm that **0.3** is the default entry.

The **Second** field specifies the second IOU threshold value (σIOU2 value).

g In the **Sigma thresholds** section, in the **Duplicate** field, delete the default entry.

The **Duplicate** field specifies the IOU duplicate threshold value (σIOU-dup value). The default value is 0.0, which means that the property is disabled.

h In the **Velocity vector frames** field, change the default entry to `11`.

This parameter defines the number of frames used to calculate the velocity vector.

i In the **Maximum undetected tracks time-to-live** field, change the default entry to `60`.

Tracks begin with multiple lives. When a track is left unmatched with any detection on a frame, it loses a life. After several frames without matching a detection, it dies. The parameter max-track-lives sets the life duration of tracks without detection.

j In the **Minimum track length for missing frames** field and the **Track retention frames** field, delete the default entries.

The **Minimum track length for missing frames** field specifies the minimum track length before permitting a missing frame. This is the tmin value. When a track has a length lower than this value, it dies as soon as it gets a missing detection.

The **Track retention frames** field specifies the number of frames that tracks keep in the history of positions in memory.

k Collapse **Tracking Properties**.

52 Configure the tracking window's coordinate properties:

a Expand **Coordinate Properties**.

b In the **Coordinate type** field, confirm that **Top left rectangle (rect)** is the default selection.

This option enables you to specify the input coordinate format. The **Top left rectangle (rect)** format specifies a bounding box by using the x and y coordinates of its top left corner along with width and height values.

c In the **Coordinate type** table, for the **X attribute** field, change the default entry for **Mapping** to `obj_%_x`.

d In the **Coordinate type** table, for the **Y attribute** field, change the default entry for **Mapping** to `obj_%_y`.

e In the **Coordinate type** table, for the **Width** field, change the default entry for **Mapping** to `obj_%_w`.

f In the **Coordinate type** table, for the **Height** field, change the default entry for **Mapping** to `obj_%_h`.

g In the **Input mappings** table, for the **Attribute count** field, change the default entry for **Mapping** to `objCount`.

h In the **Input mappings** table, for the **Attribute score** field, change the default entry for **Mapping** to `obj_%_score`.

i In the **Input mappings** table, for the **Attribute label** field, change the default entry for **Mapping** to `obj_%_label`.

53 Connect the getDetections window to the tracking window with an edge:

a Position the cursor over the anchor point at the bottom of the getDetections window so that the anchor point color changes to white.

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

The tracking window now accepts values from the getDetections window.

54 Expand **Transformations** on the **Windows** pane on the left and drag a Functional window to the workspace.

The right pane displays the Functional window's properties.

55 Specify a name for the Functional window:

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

b Collapse **Name and Description**.

56 Configure the outputTracks window's state:

a Expand **State**.

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

c Collapse **State**.

57 Configure a subscriber connector for the outputTracks window.

a Expand **Subscriber Connectors**.

b Click  .

The Connector Configuration window appears.

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

d In the **Fsname** field, enter `@DATA_DIR@/outputTracks.csv`.

e In the **Fstype** field, enter `csv`.

f Click **All Properties**.

i In the **header** property's **Value** field, select **full** from the drop-down list.

ii Click **OK**.

g Click **OK**.

58 Connect the tracking window to the outputTracks window with an edge:

a Position the cursor over the anchor point at the bottom of the tracking window so that the anchor point color changes to white.

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

The outputTracks window now accepts values from the tracking window.

59 Click the outputTracks window on the workspace.

60 Specify an output schema for the outputTracks window:

a In the right pane, click  .

b Click  .

The Output Schema window appears.

c Click  .

The Copy Fields From Input Schema window appears.

d Select all of the fields in the table with the exception of the **image** field.

e Click **OK**.

You are returned to the Output Schema window.

f Click **OK**.

61 Click 

62 Expand **Transformations** on the **Windows** pane on the left and drag a Transpose window to the workspace.

The right pane displays the Transpose window's properties.

63 Specify a name for the Transpose window:

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

b Collapse **Name and Description**.

64 Connect the outputTracks window to the transpLong window with an edge:

a Position the cursor over the anchor point at the bottom of the outputTracks window so that the anchor point color changes to white.

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

The transpLong window now accepts values from the outputTracks window.

65 Configure the transpLong window's settings:

a Expand **Settings** if it is not already expanded.

b In the **Mode** field, select **Long**.

c In the **Tag name** field, enter `obj`.

d In the **Included values** field, enter the following values:

- id
- label
- score
- x
- y
- h
- w
- center_x
- center_y
- track_x
- track_y
- vvect_x
- vvect_y

Note: Press Enter to confirm each value's entry.

e Click .

The XML Editor appears.

f Copy and paste the following XML code into the `<window-transpose>` XML tag:

```
tags-included="Object0,Object1,Object2,Object3,Object4,Object5,Object6,Object7,
Object8,Object9,Object10,Object11,Object12,Object13,Object14,Object15,Object16,
Object17,Object18,Object19,Object20,Object21,Object22,Object23,Object24,
Object25,Object26,Object27,Object28,Object29,Object30,Object31,Object32,
Object33,Object34,Object35,Object36,Object37,Object38,Object39,Object40,
Object41,Object42,Object43,Object44,Object45,Object46,Object47,Object48,
Object49,Object50,Object51,Object52,Object53,Object54,Object55,Object56,
Object57,Object58,Object59,Object60,Object61,Object62,Object63,Object64"
```

Copying and pasting the XML code above using the XML editor is quicker and more efficient that manually entering each tag using the **Included tags** field in the user interface.

Note: Ensure you paste the above XML code before the closing XML tag `/>`.

g Click .

h Collapse **Settings**.

66 Configure a subscriber connector for the transpLong window.

a Expand **Subscriber Connectors**.

b Click .

The Connector Configuration window appears.

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

d In the **Fsname** field, enter `@DATA_DIR@/transLong.csv`.

e In the **Fstype** field, enter `csv`.

f Click **All Properties**.

i In the **header** property's **Value** field, select **full** from the drop-down list.

ii Click **OK**.

g Click **OK**.

67 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.

68 Specify a name for the Filter window:

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

b Collapse **Name and Description**.

69 Configure the filterEmptyObjects window's filter settings:

a Expand **Filter**.

b In the **Filter method** field, confirm that **Expression** is selected.

c In the **Filter method** text box, enter `not isblank(id)`.

d Click  to validate the expression.

A message appears, informing you of the expression's validity.

Note: To validate your expression successfully, a working ESP server must be available. If you are using only a Kubernetes environment, skip this step as expression validation is not supported for a project running on a cluster.

e Confirm that the expression is valid.

70 Connect the transpLong window to the filterEmptyObjects window with an edge:

a Position the cursor over the anchor point at the bottom of the transpLong window so that the anchor point color changes to white.

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

The transpLong window now accepts values from the filterEmptyObjects window.

71 Expand **Transformations** on the **Windows** pane on the left and drag a Functional window to the workspace.

The right pane displays the Functional window's properties.

72 Specify a name for the Functional window:

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

b Collapse **Name and Description**.

73 Configure the outputTracks window's state:

a Expand **State**.

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

c Collapse **State**.

74 Configure a subscriber connector for the outputTracksLong window.

a Expand **Subscriber Connectors**.

b Click  .

The Connector Configuration window appears.

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

d In the **Fsname** field, enter `@DATA_DIR@/outputTracksLong.csv`.

e In the **Fstype** field, enter `csv`.

f Click **All Properties**.

i In the **header** property's **Value** field, select **full** from the drop-down list.

ii Click **OK**.

g Click **OK**.

75 Connect the filterEmptyObjects window to the outputTracksLong window with an edge:

a Position the cursor over the anchor point at the bottom of the filterEmptyObjects window so that the anchor point color changes to white.

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

The outputTracksLong window now accepts values from the filterEmptyObjects window.

76 Click the outputTracksLong window on the workspace.

77 Specify an output schema for the outputTracksLong window:

a In the right pane, click  .

b Click  .

The Output Schema window appears.

c Click  .

The Copy Fields From Input Schema window appears.

d Select the following fields:

Table 7 *A Table Displaying Output Schema Fields to Be Copied to the OutputTracksLong Window*

Window	Field	Type	Key
FilterEmptyObjects	ID	int64	Y
FilterEmptyObjects	id	int64	N
FilterEmptyObjects	label	string	N
FilterEmptyObjects	score	double	N
FilterEmptyObjects	x	double	N
FilterEmptyObjects	y	double	N
FilterEmptyObjects	h	double	N
FilterEmptyObjects	w	double	N
FilterEmptyObjects	center_x	double	N
FilterEmptyObjects	center_y	double	N
FilterEmptyObjects	track_x	array(dbl)	N
FilterEmptyObjects	track_y	array(dbl)	N

Window	Field	Type	Key
FilterEmptyObjects	vvect_x	double	N
FilterEmptyObjects	vvect_y	double	N
FilterEmptyObjects	Object_density	int32	N

e Click **OK**.

You are returned to the Output Schema window.

f Specify the **id** field to be an additional key field.

The **ID** and **id** fields are now specified as the key fields.

g Click ↑ to move the **Object_density** field to be the second field in the schema.

h Click **OK**.

78 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 `subscribers`.

e Click  below the **Connectors** label.

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

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

h Click  below the **Connectors** label.

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

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

k Click  below the **Connectors** label.

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

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

n Click  below the **Connectors** label.

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

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

q Click **OK**.

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

The Connector groups window appears.

s In the **Name** field, enter `detections`.

t Click  below the **Connectors** label.

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

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

w Click **OK**.

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

The Connector groups window appears.

y In the **Name** field, enter `stream`.

z Click  below the **Connectors** label.

aa In the Connector column, click the newly created row and select **cq1/swVideo/csv_file** from the drop-down list.

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

ac Click **OK**.

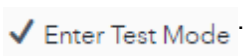
ad Configure the dependency rules. Click  below the **Dependency rules** label. After you add a row, click  again to add the next row.

Enter the following values in the rows:

Table 8 A Table Displaying the Model's Dependency Rules

Row	Controlling Group	Dependent Group
1	subscribers	detections
2	detections	stream

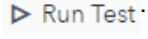
79 The model is now complete. Click  to save your model.

80 Click 

.

A new page called **Test: MOT17_tracking_proj** appears.

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

82 Click 

.

The results for each window appear on separate tabs:

- The swDetections tab shows detection data.
- The addObjid tab shows detection data with an object ID assigned to each frame in the video.
- The transpDetections tab shows transposed detection data.
- The detAgg tab shows aggregated detection data.
- The swVideo tab displays the ID of the video file.
- The getDetections tab combines a list video frames with a list of detection data.
- The tracking tab shows tracking data for each object.
- the outputTracks tab lists the model's output tracks.
- The transpLong tab shows a transposed list of the model's output tracks in long format.
- The filterEmptyObjects tab shows a filtered list of objects that do not contain empty values.
- The outputTracksLong tab shows a list of output tracks in long format.

83 To stop the test, click ■ Stop.

The project stops and then unloads from the ESP server.

Importing Models Created in SAS Model Manager into SAS Event Stream Processing Studio

Model Manager Integration Overview

Projects can reference models that are stored in the SAS Model Manager common model repository. When a project is deployed, the model is retrieved from the SAS Model Manager common model repository 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 Calculate window's input handler.

In order for a model to be retrieved successfully from the SAS Model Manager common repository and then imported for use in SAS Event Stream Processing Studio, your deployment must meet the following criteria:

- The `/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/mas-modules` directory must be owned by the same user that runs the dfesp_xml_server process (typically, this user is sas). If your deployment does not contain the file permissions specified, contact SAS Technical Support.

Note: If the `/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/mas-modules` directory does not exist, you do not need to manually create it. The directory is created automatically by SAS Event Stream Processing during the process.

- When you are importing ASTORE content on a single-server deployment or on a deployment where SAS Event Stream Processing Studio or SAS Event Stream Manager are installed on the same host as the Compute service, the following directories must exist in your deployment:
  - `/opt/sas/viya/config/data/modelsvr/astore`
  - `/models/astores/viya`

    Note: In the `/models/astores/viya` directory, `viya` is a symbolic link to the `/opt/sas/viya/config/data/modelsvr/astore` directory.

    If these directories do not exist in your deployment, contact SAS Technical Support.
- In a distributed deployment, SAS Event Stream Processing Studio or SAS Event Stream Manager (or both) can be installed on a separate server from the Compute service. If you are importing ASTORE content on a distributed deployment, create a Network File System (NFS) share on each server. Each server that hosts SAS Event Stream Processing Studio, SAS Event Stream Manager, or the Compute service requires an NFS share:
  - On each server running the Compute service, the following directories must exist:
    - `/opt/sas/viya/config/data/modelsvr/astore`

      Note: In the directory above, `astore` is a mount point to a Network File System (NFS) share.

    - `/models/astores/viya`

      Note: In the directory above, `viya` is a mount point to a Network File System (NFS) share.
  - On the server running SAS Event Stream Processing Studio or SAS Event Stream Manager, the `/models/astores/viya` directory must exist:

    Note: In the directory above, `viya` is a mount point link to a Network File System (NFS) share.

    If the directories specified above do not exist in your deployment or you do not have access to a Network File System (NFS) share, contact SAS Technical Support.

To successfully import a model created in SAS Model Manager into a specific Calculate 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 analytic store 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.

Note: Importing directly from a SAS Model Manager ZIP file is supported only for SAS Model Manager 9.x versions. If you are working in SAS Viya, this feature is not supported. For SAS Viya deployments, it is recommended that you import the model directly from SAS Model Manager. Only DS2 models are supported when importing from SAS Model Manager ZIP file. You must import DS or analytic store models directly from SAS Model Manager.

To view an example of importing a model from SAS Model Manager, see “Example Overview” on page 158.

Example Overview

This example demonstrates how to import a model created in SAS Model Manager to process input data.

Here is an example of a DS2 code file that might be imported from the SAS Model Manager common model repository 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 SAS Model Manager is installed at your deployment and that the model that you plan to import is correctly configured in it. You must also ensure that you have noted the order and type of your module’s output fields. In SAS Model Manager, assign the following variables to your model:

Name	Data Type	Input/Output Field
price	Decimal	Input
quantity	Integer	Input
security	Character	Input
time	Character	Input
tradeID	Character	Input
traderID	Integer	Input

Name	Data Type	Input/Output Field
volume	Integer	Output

Example Steps

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 `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 an ESP server is already 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 Calculate window to the workspace.

The right pane displays the Calculate window's properties.

7 Connect the Source window to the Calculate window with an edge:

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

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

The Calculate window now accepts events from the Source window.

8 Click the Calculate window in the workspace.

9 Expand **Settings**:

a In the **Calculation** drop-down list, select **User-specified**.

A **Handlers** section appears.

b Expand **Handlers** if it is not already expanded:

i Click the row that contains the Source window.

ii Click .

The Input Handler window appears.

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

iv The Import from SAS Model Manager window appears, showing SAS Model Manager repositories in a collapsed state.

v Navigate to the SAS Model Manager project that contains the model that you want to import.

vi Select the model that you want to import.

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

Note: In the **Function** drop-down list, the function that is automatically selected by default is the first function in the drop-down list. Alternatively, if the drop-down list does not contain any functions, a Score method is displayed.

To search for the model that you want to import, enter the model's name in the search bar and click .

vii Inspect the additional information about the model.

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

ix Click **OK**.

The Input Handler window appears.

Your model's XML code is updated to reference the module that you imported. The Input Handler window displays the name of the imported module and the function to call from the Calculate window.

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

xi 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: Make a note of 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 Connector Configuration 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  .

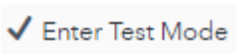
c Click  .

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.

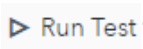
e Inspect your output schema to establish if its fields match the name, order, and type of the output fields defined in the SAS Micro Analytic Service module. Your output schema fields must also match the name, order, and type of the fields defined in your input file. These fields must match for the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

12 Click  .

13 Click 

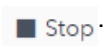
.

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:

- A `<metadata>` element that contains 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, your model's content might not be retrievable.

For a model that has been imported directly from SAS Model Manager, the model name, project name, and its version number are included in the <metadata> element.

- A <mas-modules> element that contains 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 Service 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 view only 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 Calculate windows (if the model that you imported contained additional schema fields).
- A <mas-map> element in the Calculate window that references the newly created SAS Micro Analytic Service module and its function. The <mas-map> element binds the function that is defined in the <mas-module> element to the input stream in your model's Calculate 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 SAS Micro Analytic Service Modules in SAS Event Stream Processing Studio

Overview

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

Projects can also reference models that are stored in the SAS Model Manager common model repository. When a project is deployed, the model is retrieved from the SAS Model Manager common model repository 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 Calculate window's input handler. For more information, see "Example Overview" on page 158.

Create a SAS Micro Analytic Service 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 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 analytic store 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

1 Open your project and click  .

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

3 Click  .

The Import from SAS Model Manager ZIP File 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 **Model input schema** and then copy the input schema to your clipboard. To copy the output schema for future use, click **Model 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 SAS Micro Analytic Service 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.

Working with Input Handlers

Overview

You can register event stream input handlers for the Procedural and Calculate windows. *Input handlers* process incoming event streams in your model. You can import score code created in SAS Model Manager directly into a specific Calculate window in your model. You define a SAS Micro

Analytic Service map in a Calculate window to bind a function to an input window. This binding acts as the input handler for the Calculate window.

Creating Input Handler Functions in Calculate Windows

1 Open the relevant project.

2 Click the relevant Calculate window.

The right pane displays the properties of the Calculate window.

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

4 In the **Calculation** drop-down list, select **User-specified**.

A **Handlers** section appears.

5 In **Handlers**, click the row for the input window that you want to link the import handler function to.

6 Click .

The Input Handler window appears.

7 In the **Handler type** field, select one of the following handler types:

- **SAS Micro Analytic Service** – enables you to reference a SAS Micro Analytic Service module
- **Import a module from SAS Model Manager** – enables you to reference a SAS Micro Analytic Service module
- **Import a module from SAS Model Manager ZIP file** – enables you to reference a SAS Micro Analytic Service module

The Input Handlers window reloads to display fields that relate to the handler type that you selected.

Note: In the **Function** drop-down list, the function that is automatically selected by default is the first function in the drop-down list. Alternatively, if the drop-down list does not contain any functions, a Score method is displayed.

8 Update any other fields as required.

9 Click **OK**.

The Handlers section refreshes to display the imported function.

Creating Input Handler Functions in Procedural Windows

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:

   - **Plug-in** – enables you to reference a plug-in library

     If you reference a function from a plug-in library, the function that you reference operates on a single event within an incoming event block by default. The function is repeatedly called for each event. Alternatively, you can run the function on an entire incoming event block, instead of on each event. To enable this functionality, select the **Run the function once on the incoming event block** check box.

     CAUTION
     Exercise caution when running a function on an event block instead of an event. As the function that you are running from the plug-in library is located outside of your SAS Event Stream Processing environment, this could cause your ESP server to stop processing events.

   - **DS external** – enables you to enter DATA step code directly
   - **DS external file** – enables you to reference an external file that contains 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.

   The Input Handlers window reloads to display fields that relate 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.