SAS® Event Stream Processing 5.2: Visualizing Event Streams with Streamviewer

Overview

Streamviewer is a graphical user interface that visualizes events streaming through event stream processing models. In Streamviewer, you can subscribe to individual windows on one or more event stream processing servers and display each event as a row in a table or as an element of a chart. You can save and load a collection of subscribers, their associated tables and charts, and other customized settings in Streamviewer *dashboards*.

Streamviewer dashboards are stored in a configuration database that you specify when starting Streamviewer from the command line. For the configuration database storage mechanism, you can use one of several supported enterprise databases or the stand-alone, file-based H2 database that is provided with SAS Event Stream Processing. After you have configured your storage mechanism, you can run Streamviewer from either a local or a remote installation.

Note: The following instructions assume that you have already set the environment variables DFESP_HOME and LD_LIBRARY_PATH. For more information, see *SAS Event Stream Processing on Linux: Deployment Guide* and *SAS Event Stream Processing on Windows: Deployment Guide*.

Setting Up and Running Streamviewer

Installing Streamviewer

Install Streamviewer as an optional component of your SAS Event Stream Processing deployment. For more information about installing SAS Event Stream Processing see *SAS Event Stream Processing on Linux: Deployment Guide* and *SAS Event Stream Processing on Windows: Deployment Guide*.

Streamviewer is packaged in the streamviewer-5.2.jar file, which is located in the `lib` directory of your installation directory (`$DFESP_HOME/lib` or `%DFESP_HOME%\lib`). All supported JDBC drivers are included in this JAR file.

Note: Streamviewer requires a Java Virtual Machine (JVM) of 1.7 or later.

SAS recommends that you copy the streamviewer-5.2.jar file to another location on your local machine or on another server. If you copy the JAR file to another directory, be sure to also copy the following scripts to the same location:

Linux
$DFESP_HOME/bin/streamviewer.sh

Windows
%DFESP_HOME%\bin\streamviewer.bat

The Streamviewer Configuration Database

Before you start Streamviewer, determine how you want to persistently store the Streamviewer database configuration. Streamviewer is shipped with H2, which is a file-based database engine. If you use H2 as your database, the Streamviewer configuration is stored in a file that is created when Streamviewer is invoked from the command line. For more information about H2, see http://www.h2database.com/.

Streamviewer also supports the following remote database management systems:

- PostgreSQL
- MariaDB
- MySQL
- SQLServer
- Sybase
- Oracle
- Teradata

If you choose not to use the H2 database, the remote database that you intend to use must already be running before you start Streamviewer. The database version that you use is not significant. Streamviewer supports any version of these databases.

Note: If you are moving to Streamviewer 5.2 from a previous version of Streamviewer, you must migrate your database configuration information using `dfesp_xml_client`:

```
$DFESP_HOME/bin/dfesp_xml_client -url "hostname:http_port/migrate".
```

Replace *hostname* with the host name of the server where the Streamviewer files are installed and running.

Replace *http_port* with the port number provided with the launch of the `streamviewer` script.

Database migration is not supported for Streamviewer versions earlier than 4.3.

Starting Streamviewer

Use one of the following commands to start Streamviewer. If you have copied the streamviewer-5.2.jar file to another location on your local machine or on another server, run the script from that location:

Linux
`streamviewer.sh -product database -http http_port ... additional arguments`

Windows
`streamviewer.bat -product database -http http_port ... additional arguments`

For the *database* option of the `-product` argument, use one of the following values, depending on the database that you want to use for the Streamviewer configuration:

- `h2`
- `postgres`
- `mariadb`

- mysql
- sqlserver
- sybase
- oracle
- teradata

For the *http_port* option of the -http argument, you can specify any available port number.

Use the following additional arguments as needed:

Table 1 *Command Arguments for Streamviewer Utility*

Argument	Description
-h2file *database_file*	Specifies the name of a file where you want H2 to store the configuration. The file is created if it does not already exist. Specify the database file directory with a full path or point to the appropriate relative directory. This argument is valid only for the H2 database. It is required if you are using H2.
-host *host*	Specifies the host name for the remote database. Depending on network access, you might need to specify the fully qualified domain name as well. This is required for all database products except H2.
-port *port*	Specifies the port number used by the remote database system. This is required for all database products except H2.
-db *db_name*	Specifies the name of the remote database or Oracle service to use. This argument is not used for H2, SQLServer, or Sybase. Specify the database file directory with a full path or point to the appropriate relative directory.
-user *username*	Specifies the user name required to access the remote database. This is required for all database products except H2.
-password *password*	Specifies the password required to access the remote database. This is required for all database products except H2.
-noauth	Specifies to run Streamviewer without authentication enabled. **Note:** Streamviewer uses OAuth 2.0 for user authentication and runs with authentication enabled by default.
-auth-client *client*	Specifies the value of the OAuth client. If no value is specified, the default is sas.
-auth-secret *secret*	Specifies the value of the OAuth secret. If this argument is not specified, the command line requests the OAuth client secret.
-auth-user *user*	Specifies the value of the OAuth user. If no value is specified, the default is sv.
-auth-password *password*	Specifies the value of the OAuth password. If this argument is not specified, the command line requests the OAuth password.
-ssl-cert *file*	Specifies the SSL certificate file. This argument is required if authentication is not disabled.
-ssl-password *password*	Specifies the SSL certificate password. This argument is required if authentication is not disabled.

Argument	Description
`-ssl-alias` *alias*	Specifies the SSL certificate alias. This argument is required if authentication is not disabled.
`-jar` *jarfile*	Specifies the Streamviewer JAR file and location. Defaults to **`$DFESP_HOME/lib/streamviewer-5.2.jar`** on Linux and **`%DFESP_HOME%\lib\streamviewer-5.2.jar`** on Windows. If the JAR file and `streamviewer` script are relocated to a different directory as recommended, this argument is required.

Note: You can also start Streamviewer from a Java command line. For more information, see "Starting Streamviewer from the Java Command Line".

Authentication in Streamviewer

Streamviewer uses OAuth 2.0 for user authentication and runs with authentication enabled by default. To disable authentication, use the `-noauth` parameter when starting the Streamviewer server. If `-noauth` is not specified, the SSL arguments `-ssl-cert`, `-ssl-password`, and `-ssl-alias` are required. For more information about specifying the authentication parameters when starting Streamviewer, see “Starting Streamviewer”

To create a self-signed SSL certificate, you can use `keytool`:

```
keytool -storetype PKCS12 -storepass password -alias alias -genkey -keyalg RSA -keystore full-keystore-name
 -validity 3650 -keysize 2048
```

`keytool` creates a KeyStore at *full-keystore-name*.

To generate a pseudo-random secret for the OAuth secret, you can use OpenSSL:

```
openssl rand -base64 32
```

The command line returns randomly generated Base64 output.

The Streamviewer server functions as an OAuth server. Send an HTTP request to a running Streamviewer server to retrieve an access token for OAuth authentication. You can send an HTTP request from the command line with the `curl` command-line utility:

```
curl -s -u client:secret -X POST
"https://server:port/oauth/token?grant_type=password&username=user&password=password"
```

Note: If you are retrieving a token from a server using a self-signed certificate, set the `--insecure` flag for the `curl` utility.

The command line returns a JSON object with a field called `access_token`:

```
{"access_token":"13c8edf9-4a05-41af-ba52-418c0d5f6587","token_type":"bearer",
"refresh_token":"af429a12-caa9-492b-a83d-0c21c1b28874","expires_in":43199,"scope":"user_info"}
```

In this case, the access token is `13c8edf9-4a05-41af-ba52-418c0d5f6587`. To make subsequent HTTP requests to the Streamviewer server, set your Authorization header as follows:

```
Authorization:Bearer 13c8edf9-4a05-41af-ba52-418c0d5f6587
```

Access tokens expire after one day. When a token expires, Streamviewer sends an expired token error and prompts for a new access token.

Note: Streamviewer prompts the user for an access token if a token is not provided or the token provided has expired.

Examples for Starting Streamviewer on Linux

The following examples demonstrate how to start Streamviewer using all the required parameters for each supported database product. For those commands with a specified user and password, you can substitute your own user name and password provided that you have the appropriate permissions.

H2
: Start Streamviewer on a Linux host. The H2 database file has the name **config**. If the database file **config** does not exist, the script creates the file in the current directory.

```
streamviewer.sh -product h2 -http 5990 -h2file ./config -noauth
```

MariaDB
: The user sasadmin with password saspw starts Streamviewer on a Linux host. The MariaDB database with the name **config** is running on host stream.example.com:5980.

```
streamviewer.sh -product mariadb -http 5990 -host stream.example.com -port 5980 -db ./config
-user sasadmin -password saspw -noauth
```

SQLServer
: The user sasadmin with password saspw starts Streamviewer on a Linux host. The SQLServer database is running on host stream.example.com:5980.

```
streamviewer.sh -product sqlserver -http 5990 -host stream.example.com -port 5980
-user sasadmin -password saspw -noauth
```

Oracle
: The user sasadmin with password saspw starts Streamviewer on a Linux host. The Oracle database with the name **config** is running on host stream.example.com:5980.

```
streamviewer.sh -product oracle -http 5990 -host stream.example.com -port 5980 -db ./config
-user sasadmin -password saspw -noauth
```

Examples for Starting Streamviewer on Windows

The following examples demonstrate how to start Streamviewer using all the required parameters for each supported database product. For those commands with a specified user and password, you can substitute your own user name and password provided that you have the appropriate permissions.

H2
: Start Streamviewer on a Windows host from the location where the JAR file has been relocated (***localpath***). The H2 database file has the name **config**. If the database file **config** does not exist, the script creates the file in the current directory.

```
streamviewer.bat -product h2 -http 5990 -h2file localpath\config -jar localpath\streamviewer-5.2.jar
-noauth
```

PostgreSQL
: The user sasadmin with password saspw starts Streamviewer on a Windows host. The PostgreSQL database with the name **config** is running on host stream.example.com:5980.

```
streamviewer.bat -product postgres -http 5990 -host stream.example.com -port 5980 -db .\config
-user sasadmin -password saspw -noauth
```

MySQL
: The user sasadmin with password saspw starts Streamviewer on a Windows host. The MySQL database with the name **config** is running on host stream.example.com:5980.

```
streamviewer.bat -product mysql -http 5990 -host stream.example.com -port 5980 -db .\config
-user sasadmin -password saspw -noauth
```

Sybase
: The user sasadmin with password saspw starts Streamviewer on a Windows host. The Sybase database is running on host stream.example.com:5980.

```
streamviewer.bat  -product sybase -http 5990 -host stream.example.com -port 5980
-user sasadmin -password saspw -noauth
```

Teradata
: The user sasadmin with password saspw starts Streamviewer on a Windows host. The Teradata database with the name **config** is running on host stream.example.com:5980.

```
streamviewer.bat -product teradata -http 5990 -host stream.example.com -port 5980 -db .\config
-user sasadmin -password saspw -noauth
```

Accessing the Streamviewer URL

After Streamviewer is successfully started, direct your browser to the Streamviewer URL:

```
http://hostname:http_port
```

Replace *hostname* with the host name of the server where the Streamviewer files are installed and running.

Replace *http_port* with the port number provided with the launch of the `streamviewer` script.

Connecting to Secure ESP Servers in Streamviewer

If you have set up authentication for an ESP server, you must provide authentication tokens or security credentials to Streamviewer to connect to the ESP server. When connecting to a secure server, check the **Is Secure** box of the Edit ESP Server window. Streamviewer prompts you for authentication information for the authentication that you have set up for the ESP server. For example, if you are using OAuth for authentication, an OAuth token is requested. If you are using SAS Logon, a user name and password are requested. If you are using access control, make sure that the permissions.yml file is configured correctly.

Note: If you are using a permissions file for access control, every user-object combination must have its Read and Write access explicitly defined as true or false for all engines, projects, continuous queries, and windows in a model. You can not view model objects in Streamviewer for which you do not have explicit Read access.

For more information about setting up authentication for ESP servers, see “Enabling Authentication on TCP/IP Connections” in SAS Event Stream Processing: Security.

Stopping Streamviewer

Terminate Streamviewer with the `dfesp_xml_client`:

```
dfesp_xml_client -url "http://hostname:http_port/exit"
```

Replace *hostname* with the host name of the server where the Streamviewer files are installed and running.

Replace *http_port* with the port number provided with the launch of the `streamviewer` script.

You can also terminate Streamviewer from the UNIX command line with the Ctrl + C or `kill` command.

Using Streamviewer

The Streamviewer Dashboard

Streamviewer displays events streaming through subscribed windows in tables and charts in the main area of the Streamviewer user interface, the *dashboard*.

From the pane at the top of the dashboard, you can save how you configure the elements of the dashboard to your chosen configuration database, open previously saved dashboard configurations, and export a dashboard as an XML file. You can also add new subscribers, determine how events are displayed in the dashboard, publish data directly into your subscribed windows, and create new tables or charts for saved ESP servers and subscribed windows.

Each of the icons in the dashboard pane opens a new window or performs an action:

Table 2 *Dashboard Navigation Icons*

Icon	Description
	Opens the Dashboards window, which lists the dashboards that are saved to the configuration database in use.
	Creates a new dashboard.
	Saves the current dashboard.
	Saves the current dashboard with a user-specified name.
	Reloads the current dashboard.
	Opens the ESP Servers window. Also accessible from the ESP Model Viewer window, the ESP Servers window enables you to add new servers and edit existing servers that publish events to subscriber windows in Streamviewer. For more information, see “Connecting to ESP Servers in Streamviewer”.
	Opens the ESP Model Viewer window. From this window, you can load and view an ESP model from any project, continuous query, and server connected to Streamviewer. You can also subscribe to available model windows for use in the dashboard. For more information about the ESP Model Viewer, see “Using the ESP Model Viewer”.
	Opens the ESP Subscribers window. From this window, you can add new subscribers from existing ESP servers, edit the properties of an existing subscriber, and create new tables for existing subscribers. For more information about subscribers, see “Managing Subscribers on the Dashboard”. For more information about creating charts and tables, see “Managing Tables and Charts on the Dashboard”.
	Opens the Dashboard Fields window. From the Dashboard Fields window, you can define the name and size of the columns of event data in your dashboard’s tables and charts.

Icon	Description
	Opens the Edit Dashboard window, which provides options for editing the dashboard's appearance. To add, edit, or delete display settings for fields of data that appear in the dashboard, click **Manage Fields** at the bottom of the Edit Dashboard window. From the Dashboard Fields window, you can define the name and size of the columns of event data in your dashboard's tables and charts.
	Opens your email client with a URL to your current dashboard that can be sent to someone else for viewing.
	Opens the ESP Stream Publish window, which enables you to publish directly into subscribed Source windows by listing event data in CSV format or by providing the location of a CSV data file. You can also specify the date format of your publishing stream and the event block size.
	Pauses all active subscribers in the dashboard.
	Aligns all selected charts with the selected chart farthest to the left of the dashboard. To select multiple charts, hold the Shift key and click the charts that you want to select.
	Aligns all selected charts with the selected chart farthest to the top of the dashboard. To select multiple charts, hold the Shift key and click the charts that you want to select.
	Aligns all selected charts with the selected chart farthest to the right of the dashboard. To select multiple charts, hold the Shift key and click the charts that you want to select.
	Aligns all selected charts with the selected chart farthest to the bottom of the dashboard. To select multiple charts, hold the Shift key and click the charts that you want to select.
	Opens the Import Data window. From this window, you can import dashboard configuration data from a server or a local file. For more information, see "Importing Streamviewer Data".
	Opens the Export Data window. This window displays XML code that corresponds to your current dashboard configuration. For more information, see "Exporting Streamviewer Data".
	Opens the Model Visualizer in a new tab. From the Model Visualizer, you can view an event stream processing model from a server or from an XML file. For more information about the Model Visualizer, see "Visualizing ESP Models".

Connecting to ESP Servers in Streamviewer

Streamviewer enables you to view events streaming through a model from more than one event stream processing server at a time. Before subscribing to and visualizing event streams in the dashboard, you must configure your ESP server connections.

When starting an ESP server, you must specify the HTTP port and the publish/subscribe port to connect to the server from Streamviewer. For more information about starting an event stream processing server, see *SAS Event Stream Processing: Using the ESP Server*.

Note: Streamviewer displays events for ESP servers at the same release level as Streamviewer. Connecting Streamviewer to a server running a previous version of SAS Event Stream Processing is not supported and can cause unpredictable results.

To open the ESP Servers window from the dashboard or the ESP Model Viewer window, click  on the pane at the top of the page. You can add, edit, test, and delete ESP server connections from this window.

All the server connections that you have created appear in the ESP Servers window. To edit the connection settings and server name for existing server connections, select a server and click  .

To retrieve the CPU usage statistics for event stream processing models running on an existing server, select a server and click  . A table reporting CPU usage statistics appears on the dashboard.

To monitor events flowing through a server, click  . An Event Flow Diagram appears on the dashboard.

To add an ESP server connection to the list, click  . The Edit ESP Server window appears.

Complete the following fields of the Edit ESP Server window to set up an ESP server connection:

Name
Specify a unique string for the name of the server connection.

Host
Enter the server's location.

Port
Enter the server's HTTP port.

Stats Interval
The statistics interval, measured in seconds, determines the reporting interval for the server to send CPU usage statistics to Streamviewer.

Min CPU %
Specify the minimum CPU usage percentage for windows to be reported in CPU usage statistics.

If the ESP server uses a secure connection, check **Is Secure**. For more information about connecting to a secure ESP server in Streamviewer, see "Connecting to Secure ESP Servers in Streamviewer".

If you want to verify that the information that you entered is correct and that the server is running, click  . This test can help verify the information that you entered only if the server is active.

Note: If the server is not active, you can still add the server information to the configuration for Streamviewer.

Using the ESP Model Viewer

Use the ESP Model Viewer to view event stream processing models and subscribe to windows. To open the ESP Model Viewer from the dashboard, click  .

At the top left of the ESP Model Viewer window, select the **Server**, **Project**, and **Contquery** that you want to view.

The windows in the selected continuous query and their associated edges are diagrammed in the content area.

Click  to change the orientation of the model and the appearance of the edges.

Click a window to display details about it. These details include the window's incoming and outgoing data sources and its schema fields.

You can create a subscriber to monitor a selected window in the dashboard. There are two subscriber modes:

Updating
A subscriber in Updating mode retrieves events from the server whose keys correspond to the events currently displayed in a table or chart on the dashboard.

Streaming
A subscriber in Streaming mode receives all events from the server of the subscriber's page size in intervals specified in the subscriber settings. For more information about specifying page sizes and intervals in the subscriber settings, see "Managing Subscribers on the Dashboard".

To create an Updating subscriber, select a window and click  at the top right of the ESP Model Viewer. To create a Streaming subscriber, select a window and click  at the top right of the ESP Model Viewer.

Managing Subscribers on the Dashboard

You can edit existing subscribers and add new subscribers from the ESP Subscribers window. To open the ESP Subscribers window from the dashboard, click  .

Subscribers in Streamviewer use the SAS Event Stream Processing WebSocket Publish/Subscribe API to communicate with the ESP server. For more information about the WebSocket API, see "Creating and Using the WebSocket Subscriber" in SAS Event Stream Processing: WebSocket API.

To create a new subscriber, click  . The Edit ESP Subscriber window appears. From this window, you can specify the subscriber's information up to the server level and modify the subscriber's behavior.

To edit an existing subscriber, select the subscriber and click  . The Edit ESP Subscriber window appears.

Note: You cannot change the subscriber mode, server, project, continuous query, or window of an existing subscriber.

Among the options available from the Edit ESP Subscriber window, the **Page Size** and **Delivery Interval** settings can significantly affect Streamviewer performance.

Page Size
The events displayed on the dashboard in the tables and charts connected to a subscriber make up the subscriber's *event page*. The event page size, which is measured in the number of events, determines how many events the Streamviewer client retrieves from the ESP server and displays in a table or chart on the dashboard. The default page size is 50 events.

Delivery Interval
The delivery interval, which is measured in milliseconds, determines how often Streamviewer retrieves events from the ESP server. Specifying a delivery interval can improve user interface performance for subscribed windows with large quantities of streaming events. If no delivery interval is specified, the ESP server delivers events to Streamviewer immediately after the server receives the events from the data source.

Streamviewer retrieves only as many events as the page size for each delivery interval. If, for example, the page size is 50 and the delivery interval is 500 for a given subscriber, Streamviewer will retrieve 50 events from the subscriber window every half second and display them on each table and chart that is connected to that subscriber.

To open a new table for an existing subscriber, select the subscriber and click  . For more information about editing tables and charts, see "Managing Tables and Charts on the Dashboard".

Importing Streamviewer Data

To import data directly from another running Streamviewer server or from a file, click  . The Import Data window appears. From this window, you can select whether to import data from a running Streamviewer **Server** or from a **File**.

Note: Streamviewer 5.2 only supports data imports from Streamviewer 5.2 and 5.1 servers.

If you select **Server**, complete these fields and click **Ok**:

Source Config URL
Enter the configuration URL of the server to import from. The URL is the host name and port of the Streamviewer instance.

Access Token
If the Streamviewer instance is secure, enter the access token. For more information about access tokens in Streamviewer, see “Authentication in Streamviewer”.

If you select **File**, click **Choose File** to browse for the file to upload.

Note: The imported file should have an XML structure similar to the following. This is the output from following the instructions in “Exporting Streamviewer Data”.

```
<rows continue-on-error='false'>
   <insert table='streamviewer_server'>
      <values>
         <value column='id'>e01b157a0318250a09408857a472898e</value>
         <value column='name'>bserv</value>
         ...
         </values>
   </insert>
   <insert table='streamviewer_dashboard'>
      <values>
         <value column='id'>e01ab91c-0318250a-09408857-455cf343</value>
         <value column='name'>sample</value>
         <value column='creation_time'>1485523728055</value>
         ...
         </values>
   </insert>
   ...
</rows>
```

If the Streamviewer instance is secure, enter the access token in the **Access Token** field. For more information about access tokens in Streamviewer, see “Authentication in Streamviewer”.

Exporting Streamviewer Data

Click  on the dashboard pane to export Streamviewer data to a file that can be imported to an instance of Streamviewer running on another server. The Export Data window appears. This window shows the dashboard configuration in XML format. Copy the XML configuration from the Export Data window to save it to a local file. Use the local file with the XML export data to import the data to another Streamviewer server.

Visualizing ESP Models

To open the ESP Model Visualizer in a separate tab, click  from the dashboard. The ESP Model Visualizer enables you to view models that exist on a server or in a static file.

To view a model in a local XML file, drag and drop the file into the content portion of the window.

To view a model on a running ESP server, enter the host name and port of the ESP server in the **Server** field and click  . The **Project** and **Contquery** fields are populated with the default values. You can select other continuous queries to load if they are available in the project.

Note: When entering the server host name in the **Server** field, you must specify the correct protocol.

After you load a model in the visualizer, the windows in the model and their associated edges are shown in the content area. To control the appearance of the model in the content area, modify the **Orientation** and **Link Type** settings located under the **Project** and **Contquery** fields.

Windows with a half-filled red circle in the bottom right corner are stateless and have an index of pi_EMPTY. Windows with a green check in the bottom right corner have an index other than pi_EMPTY and are stateful. Click on any window to display its XML definition in the content area at the bottom of the visualizer window.

Managing Tables and Charts on the Dashboard

After you create a new subscriber, a table of events streaming to the subscribed window appears in the dashboard. You can change the table's display type and create other visual components based on the same stream of events. You can organize data in tables and charts by key fields or by classes.

Click at the top right of a table or chart to display a menu of options. From this menu, select from the following:

Edit
Opens the Modify Chart window. From this window, you can change the type of chart that you have selected and its title and appearance.

New Chart
Opens the New Chart window. From this window, you can create a new table or chart based on the selected subscriber. This window provides the same options for tables and charts as the Modify Chart window.

Note: The types of charts available for a subscriber depend on the subscriber window's key and non-key field types. For example, series plots require subscriber windows to have numeric key fields. Time series plots require a time key field. Some chart types require the subscriber window to have numeric non-key fields.

Filter
Filters the events displayed in a table or chart. Create filters using the Expression Engine Language. For supported functions, see "Functional Window and Notification Window Support Functions" in SAS Event Stream Processing: Programming Reference.

Note: Represent floating–point numbers that are used in filters in US decimal format.

Javascript
Opens the Embed Chart Javascript window. Use the chart JavaScript to embed charts in web pages. For more information, see "Embedding Charts".

Chart Page
Opens the Chart Page URL window. From this window, you can open a web page dedicated to the selected chart in a separate tab or you can embed the chart in a web page with the chart page URL. For more information, see “Embedding Charts”.

Close
Removes the table or chart from the dashboard.

After a table or chart is displayed, you can move it by dragging it to the desired position. You can resize a table or chart by grabbing the lower right corner and dragging it. Anytime that you click on a table or chart, it appears in front of the other components on the screen.

Note: All tables and charts connected to a subscriber are synchronized. When you apply a filter to one table or chart, it affects all tables and charts connected to the same subscriber. Similarly, when you sort a table, all charts connected to the same subscriber are sorted accordingly.

Embedding Charts

Overview

Charts that are saved to a Streamviewer dashboard can be embedded in HTML web pages.

Click at the top right of a table or chart to obtain the chart JavaScript or a URL for a web page dedicated to the selected chart.

Embedding Chart JavaScript

If you want to provide more visual control of the components of the chart on a web page, you can embed the chart JavaScript in HTML. The chart JavaScript provides methods that enable you to change the appearance of embedded charts.

To view the JavaScript for the chart, click **Javascript** from the chart menu. The Embed Chart Javascript window appears.

To embed the chart JavaScript in HMTL, follow these steps:

1 Copy the following file from the SAS Event Stream Processing installation directory:

```
$DFESP_HOME/tools/svchart.tar.gz
```

2 Unpack the archive file in the directory of the page that will display the chart:

```
tar -xvzf svchart.tar.gz
```

Verify that the files are extracted to an **/esp** subdirectory relative to the page that will display the chart.

3 On the web page where you want to embed the chart, add the following code to the `head` tag:

```
<script data-main="esp/js/svchartApp" src="esp/js/libs/require.js"></script>
<link rel="stylesheet" href="esp/style/svchart.css" type="text/css" />

<script type="text/javascript">

function
setupEsp(svchart, servers)
{
   servers.setKpiSkin("modern");     //a

   servers.setGraphSkin("sheen");    //b
```

```
svchart.create("e020cb157-0318250a-09408857","http://example.com:55000").attach("mychart"); //c
</script>
```

a This portion of the `setupEsp()` function controls the appearance of embedded key performance indexes (KPIs). KPIs consist of the gauge charts available in Streamviewer. Valid arguments for the `setKpiSkin()` method include: basic, charcoal, modern, onyx, and satin.

b This portion controls the appearance of all embedded charts except gauges and tables. Valid arguments for the `setGraphSkin()` method include: sheen, crisp, gloss, matte, flat, and none.

c Replace this portion with the JavaScript code provided by the Embed Chart Javascript window. Add a unique page ID to the `attach()` method. In this example, the ID "mychart" is used.

4 Add the chart to the page using the unique page ID. In the example code here, this ID is "mychart". Typically, you should use a `div` tag to contain the embedded chart. Here is an example:

```
<div id="mychart" />
```

Embedding Charts from a Secure Streamviewer Server

If you want to embed Streamviewer charts from a secure Streamviewer server, set the authorization information in your JavaScript code.

To authenticate with a secure Streamviewer server, set the authorization for the Streamviewer chart object in the `setupEsp()` function.

```
function
setupEsp(svchart,servers)
{
        _svchart = svchart;

        svchart.setAuthorization("bearer sv_token");

        ...
}
```

Embedding Charts from an ESP Server with Authentication Enabled

If you want to embed Streamviewer charts for subscribers to an ESP server with authentication enabled, set the authentication information in your JavaScript code.

To authenticate with an ESP server with authentication enabled, define an authentication function that sets the authorization and resends the request. Then set the authenticate property in the server object to that function. For example, if the server uses OAUTH authentication, you can define a function `myauth` that sets the server authorization to an OAUTH token and then resends the request. With this function defined, you can set `servers.authenticate` to `myauth`.

```
function
setupEsp(svchart,servers)
{
        _svchart = svchart;

        servers.authenticate = myauth;

        ...
}

function
```

```
myauth(server,scheme,request)
{
        server.setAuthorization("bearer esp_token");
        request.send();
}
```

If the authentication information is valid, the request successfully authenticates, and the chart appears.