PUBLIC
SAP HANA Streaming Analytics 2.0 SP 04
Document Version: 1.0.2 – 2022-09-22

SAP HANA Streaming Analytics: Troubleshooting Guide

THE BEST RUN SAP

Content

1 SAP HANA Streaming Analytics: Troubleshooting Guide

Find workarounds for common performance, installation, and connectivity issues.

2 Locating the Streaming Version Information

When requesting technical support, provide the version and revision information for your installation of SAP HANA streaming analytics.

Context

You can find the product version information in most tools that you use with streaming analytics.

Procedure

- Using SAP HANA cockpit:
  a. On the system overview, under Platform Lifecycle Management, select **View system information**.
  b. Log in as `<sid>`adm.
  c. Select **Installed Components**, then locate SAP HANA streaming analytics in the list.
- Using SAP HANA studio:
  a. Open SAP HANA studio.
  b. In the SAP HANA Administration Console, open the **Diagnosis Files** tab.
  c. Locate the streaming analytics server log or trace file. These files have the prefix `streamingserver~`. The version information appears in the beginning of the first log file. Version information is not repeated in rollover log files.
- If you don't have access to the SAP HANA cockpit or the streaming analytics plugin for SAP HANA studio, either:
  - Refer to the streaming analytics trace files located at `/usr/sap/<SID>/HDB<instance>/<host>/trace`, or;
  - Run the `streamingclusteradmin` utility with the `--version` or `-v` option.

> **i Note**
>
> If `streamingclusteradmin` is not available, you can run most streaming analytics utilities using the `--version` option or its equivalent, excluding `streamingmonitor` and `streamingstudio`. See the *SAP HANA Streaming Analytics: Utilities Guide* for the version commands in individual utilities. All utilities from the same install have the same version number.

Related Information

SAP HANA Streaming Analytics: Utilities Guide

3 Monitoring and Analysis Tools

Use log files, studio tools, and utilities to monitor streaming analytics projects and analyze data flow.

In this section:

3.1 Understanding Server Messages

Server messages are produced to provide error and event information. Understand the pieces that make up a message to debug and monitor your system.

The following is the format of log messages:

```
[XXXXX]{-1}[-1/-1] YYYY-DD-MM HH:MM:SS.SSSSSS n Streaming UNKNOWN(-0001):[SP-X-YYYYYY](seconds)sp(pid) error message text
```

XXXXX	Thread ID of the thread generating the message.
YYYY-DD-MM HH:MM:SS.SSSSSS	UTC timestamp for the message.
n	Type of error that resulted in the message. Values include e (error), i (info), w (warning), d (debug), or f (fatal).
Streaming	Name of the component that generated the error. This will always be `Streaming` for Streaming messages.
UNKNOWN(-0001)	File and line number causing the error. This feature is not supported in Streaming log messages.
X	Logging level of the project.

YYYYYY	The module number in which the message is logged, combined with the message number. Used mostly for SAP technical support.
pid	The process ID for the Streaming Processor process.
error message text	The specific error that occurred.

3.2 Log Files

SAP HANA streaming analytics produces log files for projects and the cluster. A cluster node may contain multiple projects, each with its own project log file.

Streaming analytics stores logs in flat files. The project and cluster log files reside in the SAP HANA trace directory, `$SAP_RETRIEVAL_PATH/trace/DB_<database-name>/`. The cluster log files are named `streamingserver_<host>.3<instance-number><available internal port>.<log-serial-number>.log`.

To see your assigned internal port number, look at the `streamingserver` service in the landscape view in SAP HANA studio or run the `SELECT PORT FROM M_SERVICES WHERE SERVICE_NAME='streamingserver' AND HOST='<host name'>` statement and look at the `PORT` column in the `M_SERVICES` view. The external port number is always one number higher than the internal port. For example, if the internal port is 3xx40 then the external port is 3xx41.

Use the **Diagnosis Files** tab of the SAP HANA Administration editor to perform analysis on the logs. The timezone for these log files is the local timezone. See the *SAP HANA Administration Guide* for additional details about the diagnosis files and the Administration editor.

In this section:

Related Information

SAP HANA Administration Guide

3.2.1 Installation Log Files

Streaming analytics logs installation results, errors, and warnings from various components in different log files. Review these logs to help troubleshoot issues. If you require technical support, your representative may request that you send information from one or more of these logs.

The `streaming_suite.log` file contains a summary of the streaming analytics installation results.

The `streaming_suite.log` file and all other files are located at (Linux) `<install-dir>/log`.

Filename	Component
`streaming_suite.log`	Streaming analytics; includes a summary of installation results
`conn_lang.log`	Open Client connectivity language modules
`dbilib.log`	Open Client DB-Library
`esp_framework_install.log`	Adapter Toolkit
`esp_http_install.log`	HTTP Output Adapter
`esp_logfile_input_install.log`	Logfile Input Adapter
`esp_odbc_install.log`	ODBC Driver
`esp_pde_install.log`	PowerDesigner Extensions for streaming analytics
`esp_repserver_install.log`	Replication Server Adapter
`esp_rfc_install.log`	SAP RFC Adapter
`esp_server_install.log`	Streaming analytics server
`esp_studio_install.log`	Streaming analytics plugin for SAP HANA studio
`esp_tibco_rv_install.log`	TIBCO Rendezvous Adapter
`esp_ws_install.log`	Web Services Adapter
`espcmap.log`	Management User Interface
`jre7.log`	SAP Java Runtime Environment
`lang.log`	Open Client language modules
`open_client.log`	Open Client (OCS)
`sysam_util.log`	SySAM License Utilities

3.2.2 Project Log Files

Configure project logs to capture errors in running projects. You can configure logs for single or multiple projects in a cluster.

The streaming analytics project log and standard streams log files are both located in the SAP HANA trace directory. The standard streams file receives all output written to stdout and stderr, including SySAM licensing

information for streaming analytics, as well as messages from third party applications that write to stdout and stderr.

Project log file	`streamingserver~<workspace-name>.<project-name>.<project-instance-number>~_<machine-hostname>.<cluster-node-rpc-port>.<log-serial-number>.trc`
Standard streams log file	`streamingserver~<workspace-name>.<project-name>.<project-instance-number>~_<machine-hostname>.<cluster-node-rpc-port>.<log-serial-number>.out`
Trace directory	`$DIR_INSTANCE/<hostname>/trace`

The format of a message is: `[SP-<Severity>-<Message Code>] (<Time>) sp(<Process ID>) <Log Message>` where:

- Severity is a number which qualifies the importance of the issue. For example, 1 is a severe problem, 3 is an error, 4 is a warning, and 7 is an informational message.
- Message code helps identify the exact error message that is being printed. Since certain messages can be similar, the error code helps identify the exact issue that has occurred.
- Time is the timestamp in the server's local time.
- Process ID is a number assigned by the operating system to each running instance of an executable. Each instance gets a unique value.

Modify logging levels for projects in their project configuration files (`.ccr`), or using the Project Configuration editor in the SAP HANA Streaming Development perspective in studio. For more information about the .ccr file and the Project Configuration editor, see the *Project Configurations* section in the *SAP HANA Streaming Analytics: Developer Guide*.

To modify logging levels for a project at runtime, use `streamingprojectclient` to execute:

```
streamingprojectclient -p [<host>:]<port>/<workspace-name>/<project-name> -c
<username>:<password> "loglevel <level>"
```

Log level changes made with `streamingprojectclient` do not persist— if you restart the project without also changing the logging level in the `<project-name>.ccr` file, you lose your changes to the logging level. After you change the logging level in `<project-name>.ccr`, stop and remove the project from the node, then redeploy the project to activate the new logging level.

The timezone for these log files is the local timezone.

You can modify the size and number of log files stored in a given project using the `logfile-size` and `logfile-depth` properties in the project configuration (.ccr) file. You can edit the .ccr file manually, or by using the Project Configuration editor in the SAP HANA Streaming Development perspective in studio. For more information about the .ccr file and the Project Configuration editor, see the *Project Configurations* section in the *SAP HANA Streaming Analytics: Developer Guide*.

For information about Flex logging, see *Flex Logging* in the *SAP HANA Streaming Analytics: Developer Guide*.

Related Information

Project Configurations
Flex Logging

3.2.3 Logging Level

Logging levels range from 0 to 7, and represent a decreasing order of severity. For example, the higher the log error, the less severe the issue. The higher you set the log level, the more information you receive as errors up to and including that log level are reported. The default logging level for projects is 4.

You can set logging levels:

- In cluster configuration, logging levels apply to all projects that run on the node unless you set a different logging level in the .ccr file of a project.
- In the project configuration file, `<project-name>.ccr`.
- Using `streamingprojectclient` at runtime.

Name	Level	Description
`LOG_EMERG`	0	System is unusable
`LOG_ALERT`	1	Action must be taken immediately
`LOG_CRIT`	2	Critical conditions
`LOG_ERR`	3	Error conditions
`LOG_WARNING`	4	Warning conditions
`LOG_NOTICE`	5	Normal but significant condition
`LOG_INFO`	6	Informational
`LOG_DEBUG`	7	Debug-level messages

Adjust the log level for your streaming analytics project log file according to the type of information you want to receive in SAP HANA. The higher the trace level, the more detailed the information recorded by the trace (`streamingserver~<workspace-name>.<project-name>.<project-instance-number>~_<machine-hostname>.<cluster-node-rpc-port>.<log-serial-number>.trc`).

SAP HANA Streaming Analytics Log Level	SAP HANA Trace Level
0	f (FATAL)
3	e (ERROR)
5	w (WARNING)
6	i (INFO)
7	d (DEBUG)

3.3 Monitoring and Debugging with Studio

Use tools in studio to monitor and debug streaming analytics projects.

In this section:

3.3.1 Monitoring Project Performance

The Monitor View shows visual indicators of queue size, throughput, and CPU use for each stream and window (including local streams and windows) in a project.

Each streaming node corresponds to a stream in the model with the lines outlining the path the data flows through. The color of each node represents either QueueDepth or Rows Processed (/sec), depending on your specifications.

For example, if you select the **Color Queue Depth** option, the (Red) **Range >=** field defaults to 125, and the (Yellow) **Range >=** field defaults to 20. This means:

- If the queue depth of the streaming node is greater than or equal to 125, the node is red.
- If the queue depth of the streaming node is between 20 and 124, the node is yellow.
- If the queue depth of the streaming node is less than 20, the node is green.
- If the nodes remain white, it indicates that the monitor is not receiving data from the stream processor.

The Monitor View depicts CPU utilization as a black pie wedge in the ellipses of the node. Based on the options chosen, the remainder of the ellipses are red, yellow, or green. A fully black node represents 100 percent CPU use, based on a single CPU core. With multicore or multiprocessor environments, a fully black node may be greater than 100 percent.

To look at the performance statistics of a specific node, move your cursor over the node in the diagram.

Monitoring data is available only if the time-granularity option (otherwise known as the performance monitor refresh interval) is enabled in the CCR file. The time-granularity option specifies, in seconds, how often the set of performance records — one per stream and one per gateway connection — is obtained from the running streaming analytics project.

In this section:

3.3.1.1 Running the Performance Monitor

(Studio only) View visual indicators of queue size and CPU usage for each stream and window.

Prerequisites

The project you wish to monitor is running.

Context

You can specify a delay by changing the performance timer interval.

Procedure

1. In the SAP HANA Streaming Run-Test perspective in studio, select the server view.
2. Right-click a running project and select **Show Project Properties**.
3. In the dialog, set **Performance Monitoring Refresh Interval** to a number greater than 0.
   The default setting is 5. If the refresh interval is set to 0, the performance monitor won't run.
4. In the monitor view, click **Select Running Project**  .
5. Click **OK**.
6. Select **QueueDepth** or **Rows Processed** to specify how to color each node in the performance diagram. For either option:
   - Type in a number or use the arrow buttons in the (Red) **Range >=** field to select the range to create a red node.
   - Type in a number or use the arrow buttons in the (Yellow) **Range >=** field to select the range to create a yellow node.

   Nodes are green when they fall within a range that is not in either the (Red) Range >= or the (Yellow) Range >=.
7. Click **Zoom In** or **Zoom Out** to see a larger or smaller view of the diagram.

3.3.2 Debugging Projects in Studio

The SAP HANA Streaming Run-Test perspective in studio contains two tools for debugging data flow to assist in locating and fixing bugs within the project. These include the debugger, which allows you to set breakpoints, and the event tracer, which shows the impact of each incoming event on all streams and windows of a project.

Use the debugging tools during project development, not while streaming analytics is in production mode. Debugging features are disabled by default. Place the system in Trace mode before using the debugging features.

Studio offers an extensive suite of tools for debugging projects, but you can debug from the command line as well. See the *SAP HANA Streaming Analytics: Utilities Guide*.

In this section:

Related Information

SAP HANA Streaming Analytics: Utilities Guide

3.3.2.1 Event Tracer View

The event tracer view shows the transaction flow through the model and lets you view data in each node (stream or window).

The nodes depicted in the event tracer view are drawn as a data flow, depicting the relationships between the nodes. The function of each available button in the event tracer view is as follows:

Button	Function
Select Running Project	Presents a list of running projects available to monitor from studio.
Layout TopDown	Arranges shapes vertically for a top-to-bottom data flow.
Layout Left to Right	Arranges shapes horizontally for a left-to-right data flow.
Save	Saves the image as a JPG file.
Zoom In	Enlarges the size of the image.
Zoom Out	Reduces the size of the image.
Zoom Page	Restores the size of the image to its original size.

Button	Function
Print Performance Data to Console	Prints the collected data to the console.
Close Subscription	Closes the subscription and clears the view.
Show Current Subscription in New View	Displays the current subscription in a separate view.
Fit Shape Ids	Expands a shape to show the name of the stream or window.
Initialize With Base Data	Sends all event data from streaming analytics through the event tracer.

In this section:

3.3.2.1.1 Tracing Data Flow in the Event Tracer

Run the event tracer to trace data flow.

Prerequisites

The streaming analytics server is running.

Context

The nodes depicted in the viewer are drawn as a data flow. As a transaction is processed by each node, the color of the node changes to reflect the type of transaction.

Procedure

1. In the SAP HANA Streaming Run-Test perspective in studio, there are two methods to trace data:

Method	Procedure
Event Tracer	1. Click the event tracer view. 2. Click **Select Running Project**  to show running projects that contain streams or windows.

Method	Procedure
	3. Select a running project for the event tracer. 4. Click **OK**.
Server View	1. Select the server view. 2. To refresh the server view, click **Reconnect All**. 3. Select a running project that contains streams. 4. Right-click the project node, and select **Show in** > **Event Tracer View**.

2. Double-click a node to show the corresponding stream's data in the console view.
3. Load test data to view the impact on each stream in the event tracer. Do one of the following:
   - Click the **Upload File** tab in the toolbar below the Activate Project pane to upload data from a file.
   - In the manual input view, manually enter individual transactions by clicking the **Select Stream** icon. Select a stream. To confirm, click **OK**.

Results

The shapes in the event tracer view change color according to the status of each element.

3.3.2.1.2 Viewing the Topology Stream

The topology stream constructs the data-flow diagram, where relationships between the nodes of a project are represented as line segments.

Procedure

1. In the SAP HANA Streaming Run-Test perspective, select the **Event Tracer** view.
2. Click **Select Running Project**. Select the desired project, and click **OK**.
3. To view the entire diagram, select **Layout top down** or **Layout left to right**.
4. To view a particular node, select the section of the data-flow diagram that contains the desired stream.

3.3.2.2 Debugging with Breakpoints and Watch Variables

Use studio to control a running project by enabling tracing, pausing, resuming, and stepping of data flow through streaming analytics streams. You can also create breakpoints and watch variables on a running application.

Breakpoints are locations in the input or output of a stream or window that stop the flow of data in the streaming analytics model. A watch variable inspects the data. The function of each studio breakpoint button is as follows:

Button	Function
Trace On	Instructs streaming analytics to begin tracing (debugging). Set this parameter to use the streaming analytics breakpoint APIs.
Trace Off	Stops tracing (debugging).
Step Project	Steps through the running streaming analytics project.
Pause Project	Pauses playback for projects recorded as .rec files. When the project is paused, the records from Manual Input and File Upload cannot be updated to streams until the project is resumed.
Enable All Breakpoints	Enables all breakpoints in the list.
Disable All Breakpoints	Disables all breakpoints in the list.
Insert Breakpoint	Inserts a breakpoint item into the watch table.
Insert Watch	Inserts a watch item into the watch table.
Print Breakpoint Data to Console	Prints the breakpoint and pause state data for the current stream to the console.

Breakpoint Commands

The following breakpoint commands initiate long-running operations. Each command can be canceled before completion by clicking **Cancel Current Step**.

Button	Function
Step Quiesce from Base	Automatically steps all the derived (non-base) streams until their input queues are empty.
Step Quiesce	Automatically steps the stream and all its direct and indirect descendants until all of them are quiesced.
Step Transaction	Automatically steps until the end of transaction.
Step Quiesce Downstream	Steps the descendants of the stream but not the stream itself.

Breakpoints and watch variables persist to the workspace.

In this section:

Breakpoints [page 19]

Insert breakpoints for any stream in the project.

Watch Variables [page 20]
Insert watch variables into the watch table of the Breakpoints view in the debugger to inspect data as it flows through the project.

3.3.2.2.1 Breakpoints

Insert breakpoints for any stream in the project.

Breakpoint types include:

Local	breaks on input to the stream.
Input	breaks on a specific input stream to a stream (only Flex, join, and union can have multiple input streams).
Output	breaks when data is output from the stream.

A breakpoint can be associated with a counter (enableEvery). When a counter (n) is associated with a breakpoint, the breakpoint triggers after n events flow through the breakpoint. The counter is then reset to zero.

In this section:

Adding Breakpoints [page 19]
Add breakpoints to streams in streaming analytics.

3.3.2.2.1.1 Adding Breakpoints

Add breakpoints to streams in streaming analytics.

Prerequisites

Trace mode is enabled.

Procedure

1. In the SAP HANA Streaming Run-Test perspective, open the Debugger view.
2. Click **Trace On**.
3. Click **Insert Breakpoint**  .
4. Select the stream where you want to set a breakpoint.

5. Select the type of stream.
6. Specify when the breakpoint should trigger by entering a value in the **enableEvery** field.
7. Click **Add**.

   The selected stream appears in the table within the Insert Breakpoint dialog box.
8. Click **OK**.

   The breakpoint appears in the Debugger view within the Breakpoint table.
9. Right-click the breakpoint and select an option:
   - **Enable Breakpoint**
   - **Disable Breakpoint**
   - **Remove Breakpoint**
10. (Optional) To enable or disable all breakpoints, select either **Enable All Breakpoints** or **Disable All Breakpoints**.
11. (Optional) To remove all breakpoints, right-click within the Breakpoints table and select **Remove All Breakpoints**.
12. Click **Trace Off** to run streaming analytics.

3.3.2.2.2 Watch Variables

Insert watch variables into the watch table of the Breakpoints view in the debugger to inspect data as it flows through the project.

A watch corresponds to:

- Current input of a stream
- Current output of a stream
- Queue of a stream
- Transaction input of a stream
- Transaction output of a stream
- Output history of a stream
- Input history of a stream

Add the watches you want to monitor to the watch table before running streaming analytics. When streaming analytics runs, the watch table is updated dynamically as run-control events (run, step, pause) are sent through streaming analytics.

In this section:

Adding Watch Variables [page 21]
Add a watch element to a breakpoint.

3.3.2.2.2.1 Adding Watch Variables

Add a watch element to a breakpoint.

Prerequisites

Trace mode is enabled.

Procedure

1. In the SAP HANA Streaming Run-Test perspective, open the Debugger view.
2. Click **Trace On**.
3. Right-click in the Watch table.
4. Select **Add Watch**.
5. Select a stream from the Watch Choices box.
6. Select the type of watch you want to set up on that stream.
7. Click **Add**.
   The watch appears in the table at the bottom of the dialog box.
8. Click **OK**.
   The watch appears in the Watch table in the Debugger view.
9. To remove watches, right-click within the Watch table and select one of the following:
   - **Remove Watch** to remove a single select watch variable
   - **Remove All Watches** to remove all watch variables

3.3.2.3 Stepping Through a Project in Studio

Single-step through a project.

Prerequisites

The project is paused.

Procedure

1. In the SAP HANA Streaming Run-Test perspective, open the Debugger view.
2. In the Debugger view, click **Step Project** to perform the next step in the project.
3. Click **Cancel Current Step** to terminate the action.

3.3.2.4 Pausing SAP HANA Streaming Analytics

Pause SAP HANA streaming analytics while playing back projects with `.rec` file types.

Prerequisites

Trace mode is enabled.

Procedure

1. In the SAP HANA Streaming Run-Test perspective, open the Debugger view.
2. In the Debugger, click **Pause Project** .
3. To resume streaming analytics, click **Resume Project,** or click **Trace Off** to close the debugger.

3.4 Monitoring and Debugging with the Streaming Analytics Runtime Tool

Use views in the streaming analytics runtime tool to monitor and debug streaming analytics projects.

In this section:

3.4.1 Monitoring Streams

Monitor statistics, such as CPU usage and queue depth, for each stream and window in a running project.

Prerequisites

To monitor streams in **Cloud Streaming Workspaces**, you have the following role:

- `xs_sds_rtt_viewer_template`

In order to monitor streams in a registered server, you have the following permissions:

- `view workspace <resource-type> <resource>`
- `view project <resource-type> <resource>`

Context

You can monitor project performance through the streams monitor, which outlines the streams and windows in a project with a workflow diagram. Each shape in the diagram corresponds to a stream or window, while the arrows outline the path the data flows through. The color of each node represents either queue depth or rows processed per second, depending on your specifications.

Monitoring data is available only if the time-granularity option is enabled in the CCR file. The time-granularity option specifies, in seconds, how often the set of performance records — one per stream and one per gateway connection — is obtained from the running streaming analytics project.

Monitor settings do not get saved. If you close the monitor or the streaming analytics runtime tool, the next time you open the monitor for this project, the diagram reverts to its default configuration.

Each stream and window shows the following statistics:

Statistic	Description
Transactions per second	The number of transactions processed per second.
Rows per second	The number of rows processed per second.
Transactions in interval	The number of transactions processed since the last interval.
Rows in interval	The number of rows processed since the last interval.
Queue depth	The number of rows waiting to be processed.
Total CPU (%)	The total percentage of CPU usage for the stream.
System CPU (%)	The percentage of system CPU usage for the stream.
User CPU (%)	The percentage of user CPU usage for the stream.

Procedure

1. In the streaming analytics runtime tool, expand **Cloud Streaming Workspaces** or a registered server, then expand a workspace folder.
2. Open **Projects**.
3. Right-click on a running project and select **Monitor**.

   If the **Monitor** option is greyed out, you already have a monitor tab open, either for this project or another. Close the tab and try again.
4. In the **Monitor** tab, select **Queue Depth** or **Rows per Second**.
5. Set a range of acceptable values.
   - Set the value in the first **Range>=** field to the highest acceptable value. Any stream or window over this threshold appears red.
   - Set the value in the second **Range>=** field to a lower acceptable value. Any stream or window over this threshold, but under the value in the first field, appears yellow.

   Nodes are green when they fall under the value set for the yellow threshold.
6. (Optional) Rearrange the diagram:
   - To change the diagram's orientation, in the top right corner of the diagram, select  **Horizontal** or  **Vertical** .
   - To collapse all shapes, in the top right corner of the diagram, select  , or select  to expand them.
   - To arrange them manually, click and drag elements and connectors around.
7. (Optional) To see a list of all the events in the console, select **Print Events to Console**.

3.4.2 Debugging with the Event Tracer

Use the event tracer in the streaming analytics runtime tool to see the impact of each incoming event on every stream and window in a project. This can be useful for debugging data flow.

Prerequisites

To debug with the event tracer in **Cloud Streaming Workspaces**, you have the following role:

- `xs_sds_rtt_viewer_template`

To debug with the event tracer in a **registered server**, you have the following permissions:

- `view workspace <resource-type> <resource>`
- `view project <resource-type> <resource>`

Context

Use the event tracer during project development, not while streaming analytics is in production mode. Debugging features are normally disabled, so place the system in trace mode before using them. Nodes in the event tracer diagram are colour-coded depending on the type of operation: green for an insert, yellow for an update, and red for a delete.

You can also debug projects from studio, or from the command line. For command line tools, see Debugging Projects in Studio [page 14], and the *SAP HANA Streaming Analytics: Utilities Guide* .

Procedure

1. In the streaming analytics runtime tool, expand **Cloud Streaming Workspaces** or a registered server, then expand a workspace folder.
2. Open **Projects**.
3. Right-click on a running project and select **Event Tracer**.

   If the **Event Tracer** option is greyed out, you already have an event tracer tab open, either for this project or another. Close the tab and try again.
4. (Optional) Rearrange the diagram:

   - In the top right corner of the diagram, select  **Horizontal** or  **Vertical** to change the diagram's orientation.
   - Click and drag elements and connectors around to arrange them manually.
5. (Optional) If you don't have data going into the project, load test data to view the impact on each stream in the event tracer. Right-click on a streaming project and select **Manual Input**, then add as many rows as necessary.

3.5 Monitoring and Debugging with Utilities

SAP HANA streaming analytics includes debugging features for locating and fixing problems in your projects.

Use the debugging tools during project development, not while streaming analytics is in production mode. The debugging tools are disabled by default since they place a substantial overhead on streaming analytics.

To enable the debugging tools, run streaming analytics in trace mode.

In this section:

Pause SAP HANA Streaming Analytics [page 28]
While in trace mode, issue a pause command to pause the stream processing loop. Pausing the loop allows you to examine the stream in a static state.

The Stream Processing Loop [page 29]
The internal logic of a stream in the streaming analytics plugin for SAP HANA studio can be represented as a loop with states that correspond to ways streaming analytics handles data.

Breakpoints and Exceptions [page 30]
The breakpoint function pauses streaming analytics if there is a problem with the data model, and allows the user to troubleshoot the problem.

Notification of Debugger Events [page 32]
You can receive notifications as streaming analytics changes between running and pausing, single-steps, and when it hits breakpoints and exceptions.

Example Debugging: Pausing SAP HANA Streaming Analytics [page 32]
Use the `streamingprojectclient` command-line utility to check the pause status of streaming analytics, to pause, and to unpause.

Example Debugging: Stepping SAP HANA Streaming Analytics [page 33]
Use the `streamingprojectclient` command-line utility to advance a stream by a single step.

Example Debugging: Adding Breakpoints [page 34]
Use the `streamingprojectclient` command-line utility to add or delete breakpoints from a stream or window.

Data Examination [page 35]
View different data types produced by various streams using the `examine` command. This command only works when streaming analytics is paused.

Data Manipulation [page 38]
In `streamingprojectclient`, the `eval` command allows the debugging interface to change data within streaming analytics. The contents of global and stream local variables (including eventCache) can be changed using this functionality.

3.5.1 Trace Mode

In trace mode, streaming analytics performs extra checks for possible debugging operations and breakpoints, and collects extra information about execution history.

Use the `streamingprojectclient` utility to enable, disable, and check the status of trace mode.

For example, in an instance of `streamingprojectclient`, check the status of trace mode, turn it on, check the status again, and then turn it off:

```
streamingprojectclient> trace_mode
trace mode is off
streamingprojectclient> trace_mode on
streamingprojectclient> trace_mode
trace mode is on
streamingprojectclient>trace_mode off
streamingprojectclient>trace_mode
trace mode is off
```

Trace mode is not associated with any instance of `streamingprojectclient`. You can turn on trace mode and exit `streamingprojectclient`, and streaming analytics remains in trace mode until you turn it off in another instance of `streamingprojectclient`.

3.5.2 Step SAP HANA Streaming Analytics

Stepping lets you advance the state of streaming analytics when the system is paused.

You can advance a stream by a single step or multiple steps. To produce substantial changes, a stream may require multiple steps, so users may allow the system to run an entire transaction or run until the streams are quiesced (until all their input queues are empty). Users may also choose to run the system again, which runs until another breakpoint is triggered.

Automatic Single-Stepping

Automatic stepping ignores breakpoints or bad row exceptions. Any encountered breakpoints or bad row exceptions are reported to studio, but this does not stop the stepping.

The first auto step command, `step trans`, steps to the end of a transaction. This command does at least one common step, and then continues stepping as long as the stream stays in the COMPUTE location. The stream stops when it moves into the PUT or BAD_ROW location and allows users to examine the effect of the transaction before committing or discarding it. Call `step trans` repeatedly to step past transactions.

If the execution blocks the INPUT or OUTPUT location for longer than 0.3 seconds, `step trans` stops.

The other auto-stepping commands are related to the concept of quiescence (running the streams until they have processed all the available input). These commands are:

Command	Function
`step quiesce stream <{stream-name}>`	Automatically steps the stream and all of its direct and indirect descendants until they are quiesced.
`step quiesce downstream <{stream-name}>`	Only the descendants of a stream are stepped: the stream itself is not. Use this command to clear out the descendant streams' input queues. When the argument stream produces its output, the progression of the data through the descendant streams can be easily traced.
`step quiesce from base`	Automatically steps all derived (nonsource) streams until their input queues are empty. Use this command to clean out the queues of derived streams before processing an inconsistent record through the source stream.

> **i Note**
>
> If you exit `streamingprojectclient` or studio while streaming analytics is paused, it remains paused. When you connect with another instance of `streamingprojectclient`, it still remains paused. If you leave streaming analytics in trace mode, it remains in that mode: if it encounters a breakpoint or exception, it pauses, and stops all processing until unpaused.

You can stop streaming analytics from `streamingprojectclient` even when it is paused. The `streamingprojectclient` utility unpauses streaming analytics and disables trace mode before stopping it.

Disabling trace mode also unpauses streaming analytics.

3.5.3 Pause SAP HANA Streaming Analytics

While in trace mode, issue a pause command to pause the stream processing loop. Pausing the loop allows you to examine the stream in a static state.

While streaming analytics is in trace mode, the stream processing mechanism checks for a pause request as it enters each processing group location. Once you issue a pause command, the stream pauses and does not resume the loop until you allow it to continue. When streaming analytics is paused, no calculations happen. Any transactions in the output buffers of the stream are still consumable by subscribers. Publishing to the paused stream continues until the input buffers of the stream are full.

If the stream is engaged in actual processing when the pause is requested, processing continues until it enters the next location.

Streams are not affected by a pause request when in an I/O location. For example, the stream pauses automatically in the INPUT location if there are no transactions in the input queue, and in the OUTPUT location if the output queue is full.

The stream may move between the I/O locations to a processing location even when paused. If there is a slow subscriber on a stream, the stream's output fills up, and the stream remains in the OUTPUT location, until the buffer space becomes available. If the stream in this location receives a pause request, the request is ignored. If the stream's subscriber takes a transaction off the output buffer after this request, the stream deposits its current output transaction on the buffer and goes to the INPUT location. At this point, after entering the INPUT location, the stream recognizes the pause request. No new data is processed after the pause request, but the stream does change its location.

Pause metadata streams like any other stream. No updates should come from these streams while streaming analytics is paused, except for _ESP_RunUpdates.

Example

Use the `streamingprojectclient` command-line utility to check the pause status of streaming analytics, to pause, and to unpause:

```
streamingprojectclient> check_pause
SAP HANA Streaming
                            Analytics is not paused
streamingprojectclient> pause
streamingprojectclient> check_pause
PAUSED
streamingprojectclient> run
streamingprojectclient> check_pause
SAP HANA Streaming
                            Analytics is not paused
```

3.5.4 The Stream Processing Loop

The internal logic of a stream in the streaming analytics plugin for SAP HANA studio can be represented as a loop with states that correspond to ways streaming analytics handles data.

A normal processing sequence proceeds as follows:

1. INPUT
    1. The stream waits for the input queue to become non-empty, then picks a transaction from the head of the input queue. The transaction is visible as inTrans, the current input transaction. The transaction is processed row-by-row.
    2. The current output transaction is set to empty, prepared to collect the results of processing.
2. COMPUTE
    1. The next record is selected from the current input transaction. It becomes visible as inRow, the current input row. In some cases, the current input record may actually be two records, combined into an UPDATE_BLOCK.
    2. If this is not the first iteration of the loop, the records produced from processing the previous input record are still visible as outRow.
    3. If there are any input breakpoints defined on the stream, they are evaluated against the current input record, which may trigger streaming analytics to pause.
    4. Streaming analytics checks to see if it is paused. If so, the stream pauses here and waits for permission to continue.
    5. Finally, the actual computation is performed on the current input record. It produces zero or more output records. These records become visible as outRow, and are also appended to the end of outTrans. These records follow certain internal rules, and are not exactly the same as when they are published externally. For example, the update records at this point have the operation type UPSERT, and the delete records are SAFEDELETE.
    6. Any output breakpoints defined on the stream are evaluated against the current input record, which may trigger streaming analytics to pause.
    7. If there are more records left in the input transaction, the compute loop continues; otherwise, the stream proceeds to put the calculated data into the store, unless an exception such as division by zero has happened, in which case it proceeds to the BAD_ROW processing.
3. PUT
    1. Streaming analytics is checked to see whether paused. If so, the stream pauses here and waits for permission to continue.
    2. The new result is placed into the stream's store. The result transaction gets cleaned and transformed according to the information already in the store. The current output transaction is invisible after this point. There is no current output row. Some of these transformations are:
        - SAFEDELETEs: Either thrown away (if there was no such record in the store) or converted to DELETEs (filled with all the data that they had in the store before being deleted).
        - UPSERTs: Transformed into INSERTs or UPDATE_BLOCKs. Any remaining UPDATEs are converted to UPDATE_BLOCKs, or may be discarded if no data is changed in the record from its previous state. An UPDATE_BLOCK is a pair of records: the first one has the operation type UPDATE_BLOCK and contains the new values, while the second block has the operation type DELETE and contains the old values. When an UPDATE_BLOCK is published to the outside of streaming analytics, the second record is discarded and the first one is converted to an UPDATE. Inside streaming analytics, the entire update block is visible.

3. The PUT may also trigger an exception, for example, when trying to insert a record with a key that is already in the store. In this case, the entire transaction is aborted and the stream moves to the BAD_ROW location.
4. The current input transaction and current output transaction (already transformed) are inserted into the stream's history. The input transaction is added to the end of inHist, the output transaction appended to the end of outHist. Since the processing is done now, inTrans and inRow become invisible; outTrans and outRow are already invisible by this time.

4. OUTPUT
    1. The result transaction is queued to be published to the clients. If clients are too slow and the output buffer fills, the stream waits for buffer space to become available.
    2. The result transaction is delivered to any streams that have this stream as their input. Again, if any of their input queues become full, this stream waits for them to become available.
    3. The stream then goes to INPUT the next transaction.

Besides this main loop, there are side branches. For the streams with expiry, the EXPIRY side branch occurs every second.

3.5.5 Breakpoints and Exceptions

The breakpoint function pauses streaming analytics if there is a problem with the data model, and allows the user to troubleshoot the problem.

There are two options for pausing the system: sending an explicit pause command or setting a breakpoint on a stream or window (including local streams). You can set breakpoints on the input or output of a stream, and there is no limit to the number of breakpoints you can add. When an event hits a breakpoint, the execution is paused.

You can set specific conditions on a stream or window breakpoint. For example, you can place a filter on a breakpoint so only certain records trigger the pause. You can also set a time condition, which allows the user to trigger the breakpoint after a specified number of records.

There are two kinds of breakpoints for streams:

- **On Input**: these breakpoints are checked when a row is taken from the input transaction for processing, before any computation is performed. Streaming analytics is paused in the COMPUTE location.
    - On any input: checked for input from any stream.
    - On a particular stream: checked only when the input transaction is coming from a particular stream.
- **On Output**: these breakpoints are checked after a row has been processed. Streaming analytics is paused in the next location (COMPUTE for the next input row, PUT, or BAD_ROW).

Unconditional Breakpoints and Exceptions

A simple breakpoint is unconditional: once the stream is in the right location, the breakpoint is triggered and streaming analytics is paused. You can trigger more than one breakpoint simultaneously. Multiple breakpoints can be defined in the exact same location. Streaming analytics can tell one breakpoint from another by the unique ID assigned when the breakpoint is created. If you create two breakpoints that are the same, streaming analytics gives each one a separate ID, allowing both breakpoints to be triggered at the same time.

Once a breakpoint is created, you can enable it, disable it, or alter some of its information. The breakpoint cannot be moved to another location, or have its conditional expression changed. To make major changes, delete the breakpoint and create a new one.

You may pause streaming analytics at any record. For example, if a bug surfaces on the 1000th record passing though a stream, you can let 999 records pass, then pause and single-step from that point. To do this, configure a breakpoint to trigger on every nth row. If you restart streaming analytics, the breakpoint triggers on the 1000th row next. In `bp list`, the field `enabledEvery` shows the breakpoint number as `N`, and `leftToTrigger` shows how many records remain to be seen before the breakpoint is triggered. Every time the breakpoint is triggered, `leftToTrigger` is reset to the original value of `enabledEvery`. By default, when you create a breakpoint, `enabledEvery` is set to 1, to trigger on each row. It can be changed with the `streamingprojectclient` command `bp every`.

You can disable a breakpoint temporarily by configuring the breakpoint to trigger on every record that begins with 0, or by using the `bp on` command, which triggers the breakpoint on each record.

Conditional Breakpoints and Exceptions

Use conditional breakpoints to see a record with certain contents pass through a stream.

You can apply conditional breakpoints on streaming analytics by specifying a filter expression for a breakpoint. The filter expression is evaluated first and if it results in a false (0 or NULL) value, the breakpoint is skipped. The breakpoint is triggered only if the expression results in a true value (or its `leftToTrigger` count is reduced).

The filter expression is standard in CCLScript. It uses the data from two predefined record variables: `currow` and `oldrow`. The variable `currow` contains the current record; `oldrow` is defined only for the breakpoints on input. For INSERT and a plain UPDATE the value of `oldrow` is NULL. For UPDATE_ BLOCK, `oldrow` contains the second record of the block, the old data that is being replaced. For DELETE and SAFEDELETE `oldrow` contains the same data as `currow`. A particular field can be accessed using the usual `currow.field` syntax. You can obtain the row operation code using `getOpcode(currow)`.

The row definition that provides these predefined variables changes with different types of breakpoints.

For a breakpoint on output, the breakpoint is defined on the Row Definition. The expression is evaluated on the output rows produced during the preceding COMPUTE. Since multiple rows can be produced, the expression is evaluated on each. If no rows are produced, the expression is evaluated once with `currow` set to NULL. In this case, `oldrow` is not available because the UPDATE_BLOCK is not produced on output of COMPUTE.

For a breakpoint on input from a specific stream, the breakpoint is defined on the Row Definition of that input stream. The expression is evaluated on the record or update block that is about to be computed. Filter expressions are not permitted for this kind of breakpoint.

A source stream receives data from outside of streaming analytics instead of other streams. To add a conditional breakpoint on the input of a source stream, use the name of the source stream for the input stream with `bp add <{filter-input} {filter-input}>`.

3.5.6 Notification of Debugger Events

You can receive notifications as streaming analytics changes between running and pausing, single-steps, and when it hits breakpoints and exceptions.

To receive these updates, subscribe to the _ESP_RunUpdates stream. This stream does not retain any content; notifications bypass the stream's store, and always have the operations type UPDATE. See the *SAP HANA Streaming Analytics: Configuration and Administration Guide* for more details.

Related Information

SAP HANA Streaming Analytics: Configuration and Administration Guide

3.5.7 Example Debugging: Pausing SAP HANA Streaming Analytics

Use the `streamingprojectclient` command-line utility to check the pause status of streaming analytics, to pause, and to unpause.

Procedure

- Pause streaming analytics:

  ```
  streamingprojectclient> pause
  ```

- Run streaming analytics and continue execution:

  ```
  streamingprojectclient> run
  ```

- Check if streaming analytics is paused:

  ```
  streamingprojectclient> check_pause
  ```

  When paused, streaming analytics displays the message `PAUSED`. When not paused, it displays `streaming analytics is not paused`.

3.5.8 Example Debugging: Stepping SAP HANA Streaming Analytics

Use the `streamingprojectclient` command-line utility to advance a stream by a single step.

Procedure

Advance a stream by a single step:

```
streamingprojectclient> step [`stream`]
```

Let streaming analytics pick a stream that requires processing:

```
streamingprojectclient> step
```

In this section:

3.5.8.1 Example Debugging: Automatic Stepping

The `streamingprojectclient` command line utility offers many options for automatic stepping. These commands eliminate the need for continuous single-stepping.

Procedure

- Advance a stream to the end of a transaction:

  ```
  streamingprojectclient> step trans [`stream`]
  ```

- Step a stream and all of its direct and indirect descendants until all of their input queues are empty:

  ```
  streamingprojectclient> step quiesce stream {streamName}
  ```

- Step only the descendants of the stream until their input queues are empty:

  ```
  streamingprojectclient> step quiesce downstream {streamName}
  ```

- Step all derived streams until their input queues are empty:

  ```
  streamingprojectclient> step quiesce from base
  ```

3.5.9 Example Debugging: Adding Breakpoints

Use the `streamingprojectclient` command-line utility to add or delete breakpoints from a stream or window.

Prerequisites

Pause streaming analytics before adding a breakpoint.

Procedure

- Add a breakpoint on a stream, before it starts processing an input record from any stream:

  ```
  bp add `stream` any
  ```

- Delete the breakpoint with a specified ID:

  ```
  bp del `id`
  ```

  The ID of a breakpoint is given by using either the `bp add` or `bp list` command.

- Delete all breakpoints:

  ```
  bp del all
  ```

- Enable or disable a breakpoint with a specified ID:

  ```
  bp on|off `id`
  ```

- Enable or disable all breakpoints:

  ```
  bp on|off all
  ```

- Make the breakpoint with a specified ID trigger on every nth occasion:

  ```
  bp every `count` `id`
  ```

- Make all the breakpoints trigger on every nth occasion:

  ```
  bp every `count` all
  ```

- List all created breakpoints:

  ```
  bp list
  ```

In this section:

Example Debugging: Adding Conditional Breakpoints [page 35]

The `streamingprojectclient` command-line utility provides options for adding conditions to a breakpoint. These breakpoints trigger only when the condition evaluates at true.

3.5.9.1 Example Debugging: Adding Conditional Breakpoints

The `streamingprojectclient` command-line utility provides options for adding conditions to a breakpoint. These breakpoints trigger only when the condition evaluates at true.

Procedure

- Add a breakpoint on a stream before it begins processing an input record from another stream:

```
bp add `stream` `inputStream` [`condition`]
```

- Add a breakpoint on a stream after it has processed an input record and produced an output:

```
bp add `stream` out [`condition`]
```

3.5.10 Data Examination

View different data types produced by various streams using the `examine` command. This command only works when streaming analytics is paused.

The examination commands return the records in the same format used to send updates to common subscribers. The `streamingprojectclient` command generates records in XML format. The operation types that occur in examined data include the standard types seen by the normal subscribers, and the types that are used exclusively inside streaming analytics.

Records returned by the examination commands may be grouped in the following ways:

- Two records are grouped into an update block that is printed by `streamingprojectclient` within an XML element named <pair>.
- Multiple records and update blocks are grouped into a transaction. These transactions are printed by `streamingprojectclient` within an XML element <trans>. If a transaction contains only one record, it is printed as a single record, without the <trans> wrapping.

If a stream employs an input window, as this window fills, it begins generating SAFEDELETEs for the earlier records. To distinguish these records from the DELETEs sent by the input streams, `streamingprojectclient` includes the pseudo-field ESP_RETENTION=1 in each record.

Choose the data to examine using the following arguments of the `examine` command:

Argument	Function
kind	Determines the kind of data to be examined.
stream	Specifies the stream from which the data is taken. To receive data from all streams, leave this field empty.

Argument	Function
object	Selects the particular data unit. Use this if there are many units of that kind of data (for example, variables).

Either the stream, data, or both can be left empty or omitted if it is not applicable to the kind of data requested in the `examine` command. The kind and stream must match. You cannot request global streaming analytics data from a stream, and you cannot request per-stream data unless you specify a stream.

Some kinds of data are only available from certain streams. For example, the pattern state can be read only from a pattern stream. If the requested kind of data is not available for a certain stream, the error `No such kind of data` is returned, even if that kind of data is supported for other streams.

Some kinds of data can be used both with and without the stream argument. For example, you can use “var” without a stream to examine global variables, and with a stream to examine the variables of that stream.

The groups of data related to the input queues are heterogeneous. Each record receives a tag matching the name of the stream that produced it (records produced from source streams receive tags with the name of the source stream itself). Tags include: `queue`, `inTrans`, `inRow`, `queueHead`, `queueTail`, `inHist`, `lastInTrans`, `inHistEarliest`, and `inHistLatest`.

There are also groupings of historic data that contain a mix of input and output data. Each of these groups contains one or more pairs of transactions: the first record in each pair is an input transaction, and the second record is the matching output transaction. Each input transaction record is tagged with the name of the stream that produced it. Each output transaction record gets the tag `currow`. If any input stream records are also tagged with row, you can distinguish them by looking at the order in which they appear.

Certain kinds of data deal with history, such as the input transactions processed and outputs produced. You can obtain these data sets separately (as input history and output history) or in a combined data set. When the input and output history are examined separately, the transactions are matched by their index: the first input transaction matches the first output transaction, and so on.

The amount of historical data kept for a stream is determined by the history size setting of that stream. This setting can be set globally for all streams, or set for individual streams, using the `streamingprojectclient` command history. The default history size limit is 100 transactions. Using large history limits increases the memory usage of streaming analytics.

If trace mode is off, all history is discarded, but the limit is kept. The next time trace mode is enabled, the history begins collecting again.

A record returned by the examine `command` should be stored in an empty placeholder. If there is no ambiguity with transaction boundaries (such as with `outTrans`), `streamingprojectclient` returns no data. In other cases, such as `hist`, a placeholder shows that the transaction occurred, but produced no output. A placeholder record includes all fields, including the key fields. The value of each is set to NULL.

Some types of data records use natural field names. Records, such as "pause", return metadata. The field names in this type of record are defined in streaming analytics. Other types contain a mix of fields defined by the user and added by streaming analytics. In this type of record, the fields added by streaming analytics have the prefix `ESP_`. Do not use this prefix for user-defined fields.

In this section:

Filters [page 37]

Use filters to examine specific rows in a large volume of data. The data that is returned by the debugging commands can be refined to select only the relevant data.

Store Data [page 37]
Place store data into a file using the debugging tools on streaming analytics.

3.5.10.1 Filters

Use filters to examine specific rows in a large volume of data. The data that is returned by the debugging commands can be refined to select only the relevant data.

Data is filtered in streaming analytics before it is sent out.

The `streamingprojectclient` command `exf` performs filtering as follows:

```
exf {<kind>} [ {<stream>} [ {<object>} ] ] {<expr>}
```

The names of the stream and object are optional, similar to the command `ex`. There is an added argument containing the filter expression. The filter expression references a predefined variable, similar to the breakpoint filter expressions containing the current row. This expression compares the row with the expression and decides if the row should be returned. Whenever the filter expression returns `true (non-0, non-NULL)`, the record is displayed.

The rules for the defined variables are:

- If all rows in the returned data set are of the same type, they are wrapped in a single variable row.
- If the data kind contains rows of mixed types (the input or history data), multiple variables are defined, with names matching the XML tags printed for these records. Only one variable, matching the type of the current record, contains a value. All others are set to NULL.

3.5.10.2 Store Data

Place store data into a file using the debugging tools on streaming analytics.

This process is similar to the attribute `ofile` in the stream's element, which causes the contents of the stream to be dumped to a file when streaming analytics exits.

To dump stored data, use:

```
dump {<file-name>} {<stream-name>}
```

3.5.11 Data Manipulation

In `streamingprojectclient`, the `eval` command allows the debugging interface to change data within streaming analytics. The contents of global and stream local variables (including eventCache) can be changed using this functionality.

Data manipulation functionality works only when streaming analytics is in trace mode and paused.

The `eval` command changes data by evaluating a CCLScript statement (or block) in the stream. Only the variables are visible, not the streams or stream iterators. Any CCLScript statements are permissible, including branching and loops, but writing an infinite loop indefinitely stops streaming analytics. Temporary variables are also permissible inside the CCLScript block.

In most situations the key of the eventCache is determined by the current input record. There is no input record when using the `eval` command, so the key is not set and any operations on eventCaches have no effect. This requires the key to be set manually using the operator `<keyCache(ec-variable, record)>`. Ensure that you set this operator before performing any operations on eventCache. You may change the key more than once, even in a loop, allowing you to perform operations on multiple keys.

You may only modify the stream's local variables (those defined in the local `DECLARE` block) and global variables with the `eval` command. Variables defined inside the CCLScript blocks of a stream exist only when the appropriate methods are run and cannot be modified.

If it is possible to evaluate a unit of code, then it is not an expression, but a CCLScript statement. A CCLScript statement must either be a simple statement terminated by ";" or a block enclosed in braces "{}". Multiple statements must always be enclosed in a block. If braces are used to quote the block argument, the outside quotes do not function as the block delimiters: they are just `streamingprojectclient` quotation marks.

The following are examples of correct and incorrect blocking:

Correct Blocking

```
eval `stream` `a := 1;`
eval {a := 1;}
eval `stream` `{ typeof(input) r := [ a=9; | b= 's1'; c=1.; d=intseconddate(0);];
        keyCache(s0, r); insertCache(s0, r); }`
eval {stream} {{ typeof(input) r := [ a=9; | b= 's1'; c=1.; d=intseconddate(0);];
        keyCache(s0, r); insertCache(s0, r); }}
```

Incorrect Blocking

```
eval `stream` `a := 1`
eval {a := 1}
eval `stream` `typeof(input) r := [ a=9; | b= 's1'; c=1.; d=intseconddate(0);];
        keyCache(s0, r); insertCache(s0, r);`
eval {stream} { typeof(input) r := [ a=9; | b= 's1'; c=1.; d=intseconddate(0);];
        keyCache(s0, r); insertCache(s0, r); }
```

4 Performance and Tuning Tips

Optimizing performance in SAP HANA streaming analytics requires tuning at the project level as well as at the infrastructure level (machine, OS, network configuration, and so on).

If you tune your projects to produce maximum throughput and minimum latency, but do not configure your infrastructure to handle the throughput, you will experience sub-optimal performance. Likewise, if you configure your infrastructure to handle maximum throughput, but do not tune your projects, your performance suffers.

In this section:

cockpit. If the monitoring tool shows one stream in the project using the CPU more than other streams, refine the project to ensure that the CPU is used evenly across the streams.

Improving Aggregation Performance [page 53]
Aggregation functions typically require the server to iterate over every element in a group. For this reason, the performance of the aggregation operator is inversely proportional to the size of the group.

Switch from Decimal to Money Datatype [page 55]
Using the `money` datatype rather than the `decimal` datatype can improve the performance of a project.

Recompiling Streaming Projects [page 56]
Recompile existing streaming projects that were compiled with earlier compiler versions to take advantage of the latest enhancements.

Improving Studio Performance [page 56]
When you have multiple projects running, studio can slow down. You can improve performance by adjusting preferences in the SAP HANA Streaming Run-Test perspective.

4.1 Distributing Load through Parallelization

To improve performance of large streaming analytics projects, separate the data into smaller chunks that are processed within their own partitions. Processing on multiple partitions in parallel can improve performance over processing in one large partition.

There are various ways to parallelize your streaming analytics project.

1. Application-based Partitioning

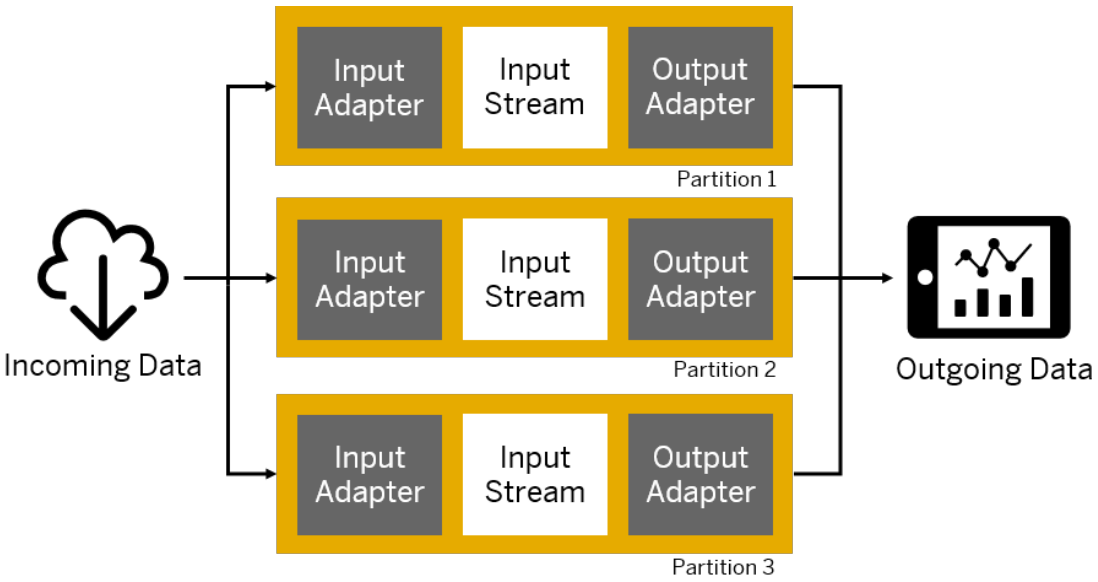
You can send all incoming data to each of the input adapters within your streaming analytics project, and then attach each of these adapters to a stream that filters a subset of the overall incoming data. The output adapters receive this data and output it to the external datasource.

Advantages:

- Improves performance and processes high volumes of data since having multiple streams processing subsets of the data divides the load on the processor.
- No need to create a custom adapter or do any custom coding aside from specifying the filtering.
- Allows for partitioning across cores, but this type of partitioning is best suited across machines.

Disadvantages:

- Must duplicate the input data feeding into the input adapters.



Application-based Partitioning

2. Partitioning Using a Custom Adapter

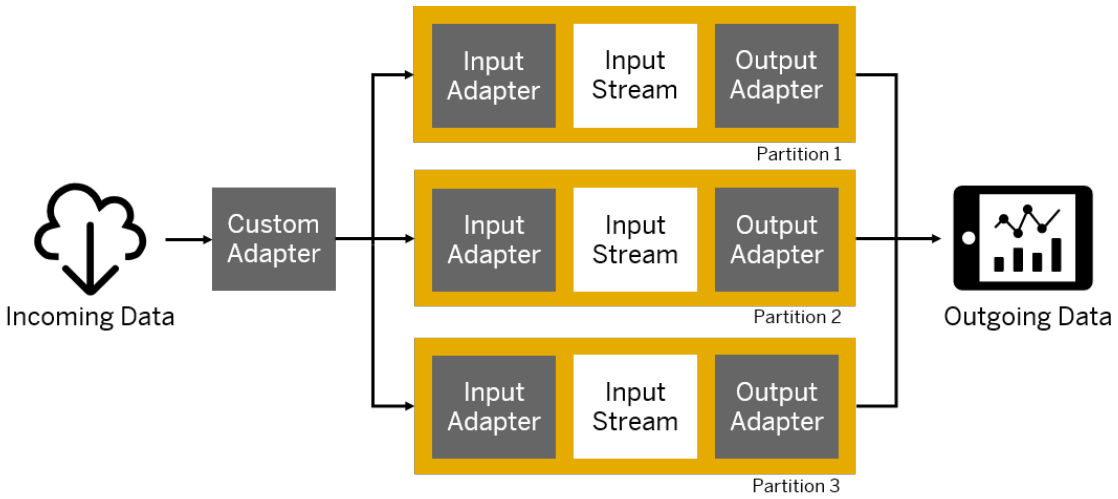
You can write a custom adapter to receive input data and publish it to various streams, keyed streams, or windows on separate machines. These streams or windows process and send this data to separate output adapters, which then publish it to the end datasource. The custom adapter is responsible for partitioning the input data in this scenario.

Advantages:

- Improves performance and processes high volumes of data by filtering incoming data across multiple machines.
- Adapters are customizable to meet partitioning requirements.
- No need to duplicate any data.
- Allows for partitioning across cores, but this type of partitioning is best suited across machines.

Disadvantages:

- Requires more effort in terms of coding because you create a custom adapter. You cannot currently partition the available adapters provided with streaming analytics.



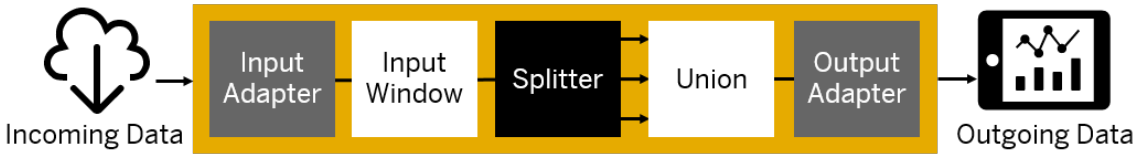
Partitioning Using a Custom Adapter

3. Partitioning Using a SPLITTER Statement

You can use the CCL SPLITTER object to subdivide input data based on specific criteria, and then a UNION statement to consolidate the data before sending it to the output adapter.

Advantages:

- More flexibility in the operations that you can perform on streams. For example, you first split the data, perform operations on the resulting streams, and then consolidate the data again.
- Allows for partitioning across cores.



Partitioning Using SPLITTER and UNION Statements

Although the example in the illustration uses a single input adapter, you can use a SPLITTER when using multiple input adapters.

> **i Note**
>
> Using the JOIN object does not provide the same performance benefit as using UNION. In fact, the JOIN operation can degrade performance considerably, so to optimize performance, parallelizing your project using UNION is recommended over using JOIN.

In both cases, the number of parallel instances is limited to the throughput of the UNION and the SPLITTER, when used. In addition, the number of parallel instances depends on the number of available CPUs.

4. Automatic Partitioning

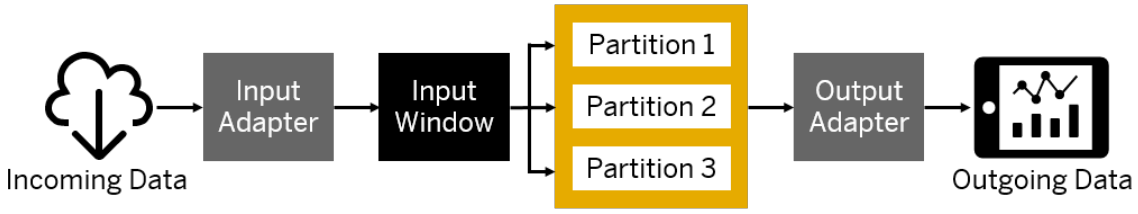
You can create multiple parallel instances of a given element (keyed stream, stream, window, module) and partition input data into these instances. Partitioning data this way results in higher performance as the workload is split across the parallel instances. If using this scenario, you can partition the CCL elements using CUSTOM, HASH, or ROUND ROBIN partitioning.

Advantages:

- Ideal for complex projects that perform computationally expensive operations, such as aggregation and joins.
- Easy to add to a project.
- Allows for partitioning across cores.

Disadvantage:

- Lacks the ability to order output.



Automatic Partitioning

General Guidelines

Hash partitioning uses hash functions to partition data. The hash function determines which partition to place a row into based on the column names you specify as keys. These do not have to be primary keys. Round-robin partitioning distributes data evenly across partitions without any regard to the values.

Choose a type based on the calculations you are performing on the input data. For example, round-robin is sufficient for stateless operations like simple filters, but not for aggregation as this would produce differing results. Hash partitioning is necessary for grouping records together, but grouping may not evenly distribute the data across instances.

When implementing the preceding scenarios, use round-robin or key-based partitioning. Round-robin partitioning provides the most even distribution across the multiple parallel instances, but is recommended only for projects limited to insert operations (that is, no updates or deletes). For projects using insert, update, and delete operations, key-based partitioning is preferable. Any update or delete operation on a record should occur on the same path where the record was inserted, and only key-based partitioning can guarantee this. However, if the HASH function is not applied correctly, key-based partitioning can distribute load unevenly, resulting in some partitions having a higher burden than others.

For more information on the SPLITTER and UNION statements, see the *SAP HANA Streaming Analytics: CCL Reference*.

Related Information

SAP HANA Streaming Analytics: CCL Reference

4.2 Distributing Load through Modularization

You can optimize performance by breaking projects into modules. This strategy spreads the load out to more cores, increasing throughput.

Use modules to double, quadruple, and so on, the number of partitions, with little additional code. The more partitions you create, the more you distribute the load.

For more information, see the *Modularity* section in the *SAP HANA Streaming Analytics: Developer Guide*, and the *SAP HANA Streaming Analytics: CCL Reference*.

Related Information

Modularity
SAP HANA Streaming Analytics: CCL Reference

4.3 Streaming Data Flow

The throughput of the streaming analytics project depends on the throughput of the slowest component in the project.

Each stream in streaming analytics has an internal queue that holds up to 1024 messages; this queue is composed of multiple internal queues. The queue size is hard-coded and cannot be modified. An internal queue buffers data feeding a stream if that stream is unable to keep up with the incoming data.

Consider an example where data flows from an input adapter, through streams A, B, and C, and then through an output adapter. If the destination target of the output adapter cannot handle the volume or frequency of messages being sent by the output adapter, the internal queue for the stream feeding the output destination fills up, and stream C cannot publish additional messages to it. As a result, the internal queue for stream C also fills up and stream B can no longer publish to it.

This continues up the chain until the input adapter can no longer publish messages to stream A. If, in the same example, the input adapter is slower than the other streams, messages will continue being published from stream to stream, but the throughput is constrained by the speed of the input adapter.

If your output destination is a database, you can batch the data for faster inserts and updates. Set the batch size for a database adapter in the data service for the cluster. For information on managing data services, see the *Configuring External Database Access* section in the *SAP HANA Streaming Analytics: Developer Guide*.

Batching data carries some risk of data loss because the database adapters run on an in-memory system. To minimize the risk of data loss, set the batch size to 1.

Related Information

Configuring External Database Access

4.4 Log Store Considerations

The size and location of your log stores can impact performance.

Sizing the log stores correctly is important. A store that is too small requires more frequent cleaning cycles, which severely degrades performance. In the worst case, the log store can overflow and cause the processing to stop. A store that is too large also causes performance issues due to the larger memory and disk footprint. For detailed information on calculating the optimal log store size, see *Sizing a Log Store* in the *SAP HANA Streaming Analytics: Developer Guide*.

When storing streaming analytics data locally using log stores, use a high-speed storage device (for example, a raid array or SAN, preferably with a large dynamic RAM cache). Putting the backing files for log stores on single disk drives (whether SAS, SCSI, IDE, or SATA) always yields moderately low throughput.

Related Information

Sizing a Log Store

4.5 Batch Processing

When stream processing logic is relatively light, inter-stream communication can become a bottleneck. To avoid such bottlenecks, you can publish data to the streaming analytics server in micro batches. Batching reduces the overhead of inter-stream communication and thus increases throughput at the expense of increased latency.

Streaming analytics supports two modes of batching: envelopes and transactions.

- Envelopes – When you publish data to the server using the envelope option, the server sends the complete block of records to the source stream. The source stream processes the complete block of records before forwarding the ensuing results to the dependent streams in the graph, which in turn process all the records before forwarding them to their dependent streams. In envelope mode, each record in the envelope is treated atomically, so a failure in one record does not impact the processing of the other records in the block.

- Transactions – When you publish data to the server using the transaction option, processing is similar to envelope mode in that the source stream processes all of the records in the transaction block before forwarding the results to its dependent streams in the data graph. Transaction mode is more efficient than envelope mode, but there are some important semantic differences between the two. The key difference between the two modes is that in transaction mode, if one record in the transaction block fails, then all records in the transaction block are rejected, and none of the computed results are forwarded downstream. Another difference is that in transaction mode, all resultant rows produced by a stream, regardless of which row in the transaction block produced them, are coalesced on the key field. Consequently, the number of resulting rows may be somewhat unexpected.

In both cases the number of records to place in a micro batch depends on the nature of the model and needs to be evaluated by trial and error. Typically, the best performance is achieved when using a few tens of rows per batch to a few thousand rows per batch. While increasing the number of rows per batch may increase throughput, it also increases latency.

4.6 Main Memory Usage

There are no SAP HANA streaming analytics configuration settings that directly set up or control RAM usage on the machine. However, a reference from streaming analytics counts records in the system, ensuring that only one copy of a record is present in memory, although multiple references to that record may exist in different streams.

Memory usage is directly proportional to the number of records in a project. To limit the amount of memory the entire instance of streaming analytics uses before it reports an out-of-memory condition, use the `ulimit` command to restrict the amount of memory available to each shell process.

4.7 Monitor Project Memory Usage

When the streaming analytics server is running at log level INFO and it is shut down cleanly, it reports the amount of memory consumed by various project components, adding this information to the streaming analytics project log file. You can also generate this report on-demand without shutting down.

The log level is a project configuration option on the **Advanced** tab of the Project Configuration editor in studio. If the level is set at 6 when the streaming analytics server shuts down, a report detailing project statistics is printed.

The files generated by a project, including the project log file, are placed in the SAP HANA trace directory. Set the log level to 6 to generate a report when shutting down. To change the log level at run time, use the `streamingprojectclient` tool and execute:

```
streamingprojectclient -p [<host>:]<port></workspace-name/project-name> -c
<username>:<password> "loglevel 6"
```

Alternatively, use the `streamingprojectclient` command `get_memory_usage all` to generate this report without having to set the log level or shut the project down. The report measures input and output

streams, queues, adapters, and reference caches for their memory use. Use this data to identify memory usage bottlenecks, and fine-tune your project layout to improve throughput.

> **i Note**
>
> In some cases when using `get_memory_usage all`, some rows of data that are shared by components are subsequently recorded more than once in the report. Rows in a gateway queue, for example, are likely to be in a store, and possibly an input queue as well. Cross-referencing between component reports can help identify where select rows are recorded, but the overall report nonetheless serves to highlight where excessive memory is being used.

When looking for component-specific data, qualify the component type and name in the project being tracked. Component types include:

- gateway
- stream or window
- global
- reference cache

Component names include:

- all
- `<client-ID><client-name>`
- `<stream-name>`
- `<adapter-name>`
- `<reference-name>`

> **i Note**
>
> While running this report, you may notice a degradation in performance until the report is complete.

In this section:

4.7.1 Gateway Memory Usage

Gateway establishes queues where it stores incoming data and, depending on the type of subscription, creates additional queues, indices, and stores for aggregation. When requesting a report for gateway, you can request data for all client connections, specify a client IP address, or specify an IP address and name pairing.

Use the `streamingprojectclient get_memory_usage` command to view all gateway clients:

```
get_memory_usage gateway
```

Use the `streamingprojectclient get_memory_usage` command to view gateway clients on a specific IP:

```
get_memory_usage <IP-Address>
```

Use the `streamingprojectclient get_memory_usage` command to view one specified client:

```
get_memory_usage <IP-Address><Client-ID>
```

A gateway report includes the following:

```
<Client-IP><Client-ID>:
Input Queue:
<subscriber-queue-memory-size> bytes in <subscriber-queue-number-of-records>
records
Aggregation Indices:
<aggregation-index-memory-size> bytes
Order by Store:
<order-by-memory-size> bytes in <number-of-records> records
Output Queue:
<output-queue-memory-size> bytes in <output-queue-number-of-rows> rows
```

The following example illustrates the report when using a specified client IP address and client ID:

```
[SP-6-131096] (71.084) sp(14723) Gateway Client 10.7.168.66 (129) Memory usage:
[SP-6-131094] (71.084) sp(14723) Queue: 0 bytes in 0 records
[SP-6-131097] (71.084) sp(14723) Aggregation Indices: 12,902 bytes in 26 records
[SP-6-131098] (71.084) sp(14723) Result Set Store: 1,846 bytes in 26 records
[SP-6-131095] (71.084) sp(14723) Gateway Output Store: 0 bytes in 0 rows
```

Considerations for gateway memory usage:

- **Queues:** You can adjust the settings of your connection tool to alleviate any strain on the connection. Increase the amount of records your tool can consume, or send fewer records by filtering data into other streams and subscribing to those streams. See the *SAP HANA Streaming Analytics: Configuration and Administration Guide* for information on project configurations.
- **Aggregation Indices:** This is a temporary index that is only stored when specified in the query to the client connection. This memory usage ends when the query is finished.
- **Result Set Store:** Like aggregation indices, this memory usage is temporary.
- **Output Store:** This is a concern only if the subscription is pulsed, where records are collected and delivered at set intervals. If the output store is holding onto too much data, lower the pulse subscribe interval. See the *SAP HANA Streaming Analytics: Utilities Guide* for information on controlling the pulse rate.

Related Information

SAP HANA Streaming Analytics: Configuration and Administration Guide
SAP HANA Streaming Analytics: Utilities Guide

4.7.2 Stream and Window Memory Usage

When requesting a report for streams, request data for all streams in a project or a specific stream by name.

There are different kinds of streams to monitor. Most allocate a temporary amount of memory to process rows, and some streams keep rows and other data for the entire project lifecycle. Streams can also have CCLScript local declare block variables of basic data types, records, dictionaries, vectors, xml values, and event caches.

Use the `streamingprojectclient get_memory_usage` command to view the memory usage for all streams:

```
streamingprojectclient get_memory_usage stream
```

Use the `streamingprojectclient get_memory_usage` command to view memory usage for a single stream:

```
streamingprojectclient get_memory_usage stream '<stream-name>'
```

A report includes the following:

```
<Stream-Name>:
Store:
<store-memory-size> bytes in <number-of-stored-rows> records stored
Input Queue:
<Input-Queue-memory-size> bytes in <number-of-queued-rows> records
Transaction Export Queue:
<Transaction-Queue-memory-size> bytes in <number-of-queued-rows> records
Aggregation Index:
<Aggregation-Index-memory-size> bytes in aggregation index
Guaranteed Delivery Store:
<GD-Store-memory-size> bytes in <number-of-stored-rows> records
```

Considerations for stream or window memory usage:

- **Store:** A log store that is too large can hinder performance due to larger disk and memory requirements. To reduce memory usage, consider adjusting your retention policy on the store to retain less data.

  > **i** Note
  >
  > Log stores that are too small can cause processing to stop due to overflow. They can also cause significant performance degradation due to frequent cleaning cycles. See *Creating a Log Store* in the *SAP HANA Streaming Analytics: Developer Guide* for information regarding log store optimization.

- **Input Queue:** Examine other objects that are consuming messages from the stream, and determine if their queues are full. If a queue is not full, check the object feeding into it for potential bottlenecks. Reduce the messages coming into the stream by filtering and distributing the data through multiple streams.

- **Transaction Export Queue:** Huge transactions are the primary cause of excessive memory use. Avoid constructing a transaction with an excessive number of records. Determine if any secondary objects like tables are joined into the row without any equality conditions.
- **Aggregation Index:** If a stream feeds input to an aggregation window directly, the memory usage of the aggregation index increases without bound. To prevent such unbounded growth, insert an intermediate window between the stream and the aggregation window. Consider lowering the retention policy of the store or unnamed window that has the aggregation.

  > **i** Note
  >
  > There are advantages and disadvantages to changing the retention policy. See *Aggregation* in the *SAP HANA Streaming Analytics: Developer Guide* for more information.

- **GD Mode:** A stream with guaranteed delivery (GD) mode enabled stores records until all rows are fully processed. If GD Mode is enabled, but not required, disable it.

Depending on which CCLScript variables exist in the stream, a report includes:

```
CCLScript Variables:
<Stream-Dictionary-size> in bytes
<Stream-Event-Cache-size> in bytes
<Stream-Vector-size> in bytes
<Stream-Records-size> in bytes
<Stream-XML-Values-size> in bytes
<Primitive-Variables-size> in bytes
```

Considerations for CCLScript local declare block variables:

- Multiple variables of the same type will not be grouped together.
- The sizes of your variables are dependent on their usage. For more information on declare block variables, consult the *CCL Statements* section of the *SAP HANA Streaming Analytics: CCL Reference*.

The following example illustrates a report when tracking streams for memory usage:

```
SP-6-124001] (191.065) sp(21115) Log store: 3,692 bytes in 52 records
[SP-6-124001] (191.065) sp(21115) Input queue: 700,000 bytes in 46,100 records
[SP-6-124001] (191.065) sp(21115) Export queue 1,000,000 bytes in 67,040 records
[SP-6-124001] (191.065) sp(21115) Aggregation Index: 1,545,000,000 bytes in
aggregation index
[SP-6-124001] (191.065) sp(21115) GD Store: 0 bytes in 0 records
[SP-6-124001] (191.065) sp(21115) StreamDic1: 3 bytes
[SP-6-124001] (191.065) sp(21115) StreamEvent1: 28,668 bytes
[SP-6-124001] (191.065) sp(21115) StreamVec1: 19 bytes
[SP-6-124001] (191.065) sp(21115) StreamVec2: 78 bytes
[SP-6-124001] (191.065) sp(21115) StreamRec1: 111 bytes
[SP-6-124001] (191.065) sp(21115) StreamRec2: 111 bytes
[SP-6-124001] (191.065) sp(21115) StreamXml1: 72 bytes
[SP-6-124001] (191.065) sp(21115) Primitive Variables: 32 bytes
```

Related Information

Creating a Log Store
Aggregation
CCL Statements

4.7.3 CCLScript Variables Memory Usage

When requesting information on CCLScript memory usage, specify your search for either global or local declare blocks.

CCLScript variables include basic datatypes, records, dictionaries, vectors, xml values, and event caches. These variables are declared in DECLARE statements, which help to define a project's computations and logic. Local declare blocks are used in regular and Flex streams, while global declare blocks are available to an entire project.

A report for CCLScript variables can include the following:

```
<Stream-Dictionary-size> in bytes
<Stream-Event-Cache-size> in bytes
<Stream-Vector-size> in bytes
<Stream-Records-size> in bytes
<Stream-XML-Values-size> in bytes
<Primitive-Variables-size> in bytes
```

Use the following `streamingprojectclient` commands to report CCLScript variables present in local streams or a single stream:

```
streamingprojectclient get memory usage stream
```

```
streamingprojectclient get_memory_usage stream '<stream-name>'
```

The following example illustrates the report when tracking CCLScript variable memory use in streams:

```
[SP-6-124001] (191.065) sp(21115) streamDic1: 3 bytes
[SP-6-124001] (191.065) sp(21115) streamEvent1: 28,668 bytes
[SP-6-124001] (191.065) sp(21115) streamVec1: 19 bytes
[SP-6-124001] (191.065) sp(21115) streamVec2: 59 bytes
[SP-6-124001] (191.065) sp(21115) streamVec3: 13 bytes
[SP-6-124001] (191.065) sp(21115) streamRec1: 111 bytes
[SP-6-124001] (191.065) sp(21115) streamXml1: 72 bytes
[SP-6-124001] (191.065) sp(21115) Primitive Variables: 32 bytes
```

> **i Note**
>
> When using `stream` and `stream <stream-name>` keywords for your report, other components of the stream are reported. See *Stream and Window Memory Usage*.

Use the `streamingprojectclient` command to report CCLScript variables in global declare blocks:

```
streamingprojectclient get_memory_usage global
```

The following example illustrates the report when tracking CCLScript variables' memory use in global declare blocks:

```
[SP-6-124001] (191.065) sp(21115) globalDic1: 64 bytes
[SP-6-124001] (191.065) sp(21115) globalRec1: 111 bytes
[SP-6-124001] (191.065) sp(21115) globalRec2: 311 bytes
[SP-6-124001] (191.065) sp(21115) globalRec3: 245 bytes
[SP-6-124001] (191.065) sp(21115) globalVec1: 66 bytes
[SP-6-124001] (191.065) sp(21115) globalVec2: 78 bytes
[SP-6-124001] (191.065) sp(21115) globalXml1: 72 bytes
[SP-6-124001] (191.065) sp(21115) Primitive variables: 32 bytes
```

Considerations for CCLScript declare block variables:

- Multiple variables of the same type are not grouped together.
- The sizes of your variables are dependent on their usage. For more information on declare block variables, consult the *CCL Statements* section of *SAP HANA Streaming Analytics: CCL Reference*.

Related Information

Stream and Window Memory Usage [page 49]
CCL Statements

4.7.4 Reference Cache Memory Usage

References can store information in potentially unlimited-size caches. When requesting a report for memory allocated to the reference cache, specify the reference name.

Use the `streamingprojectclient get_memory_usage` command to view the size of a reference cache:

```
get_memory_usage <reference-name>
```

A report includes the following:

```
<reference-name>:
<cache-size> bytes in <number-of-cached-queries> queries
```

Considerations for reference memory usage:

- Change the retention policy on your reference by lowering the `maxCacheSize` parameter. This is the only way to reduce memory consumption for this component.

  > **i Note**
  >
  > There are significant advantages and disadvantages to adjusting this parameter. See *Working with Reference Table Queries* in the *SAP HANA Streaming Analytics: Developer Guide*.

Related Information

Working with Reference Table Queries

4.8 CPU Usage

SAP HANA streaming analytics automatically distributes its processing load across all the available CPUs on the machine. If the processing of a data stream seems slow, monitor the CPU utilization of each stream using either the `streamingmonitor` utility from the command line or through SAP HANA cockpit. If the monitoring tool shows one stream in the project using the CPU more than other streams, refine the project to ensure that the CPU is used evenly across the streams.

The queue depth is also an important metric to monitor. Each stream is preceded by a queue of input records. All input to a given stream is placed in the input queue. If the stream processing logic cannot process the records as quickly as they arrive to the input queue, the input queue can grow to a maximum size of 1024 records. At that point, the queue stops accepting new records, which results in the automatic throttling of input streams. Since throttled streams require no CPU time, all CPU resources are distributed to the streams with the full queues, in effect performing a CPU resource load balance of the running project. When the input stream of a queue is blocked but the stream has managed to clear half of the pending records, the queue is unblocked, and input streams can proceed to supply the stream with more data.

If this inherent load balancing is insufficient to clear the input queue for any given stream, the backup of the queue can percolate upward causing blockages all the way up the dependency graph to the source stream. If your monitoring indicates growing or full queues on any stream or arc of streams in the directed graph, examine this collection of streams to determine the cause of the slow processing.

4.9 Improving Aggregation Performance

Aggregation functions typically require the server to iterate over every element in a group. For this reason, the performance of the aggregation operator is inversely proportional to the size of the group.

Aggregation functions can be used in a SELECT statement along with a GROUP BY clause, or over event caches in CCLScript inside UDFs and Flex operators.

For the `sum()`, `count()`, `avg()`, and `valueInserted()` aggregation functions, the server can perform additive optimization, where the function executes in constant time. In such cases, the time it takes to perform an operation is the same regardless of group size.

In a SELECT statement, the server performs additive optimization, provided that functions eligible for optimization are used in all values being selected, except for the columns referenced in the GROUP BY clause.

The following SELECT statement is optimized for additive optimization since all non-GROUP BY columns (`name`, `counter`, and `summary`) only use additive aggregation functions (that is, `valueInserted()`, `sum()`, and `count()`).

```
CREATE OUTPUT WINDOW AggResult
    SCHEMA (id INTEGER, name STRING, counter INTEGER, summary FLOAT)
    PRIMARY KEY DEDUCED
AS
    SELECT BaseInput.intData_1 AS id,
        valueInserted(BaseInput.strData_1) AS name,
        count(BaseInput.intData_1) AS counter,
        sum(BaseInput.dblData_1) AS summary
FROM BaseInput
```

```
GROUP BY BaseInput.intData_1;
```

> **i Note**
>
> For optimal performance, when selecting only the column in a SELECT statement with a GROUP BY clause, use the valueInserted function, where feasible.

The following SELECT statement is not optimized for additive optimization since one of the non-GROUP BY columns (`name`) directly selects a column that cannot be computed additively.

```
CREATE OUTPUT WINDOW AggResult
    SCHEMA (id INTEGER, name STRING, counter INTEGER, summary FLOAT)
    PRIMARY KEY DEDUCED
AS
    SELECT BaseInput.intData_1 AS id,
        BaseInput.strData_1 AS name,
        count(BaseInput.intData_1) AS counter,
        sum(BaseInput.dblData_1) AS summary
FROM BaseInput
GROUP BY BaseInput.intData_1;
```

When applying aggregation functions over an event cache, additive optimization is turned on when using the `sum()`, `count()`, `avg()`, or `valueInserted()` functions only in the ON clause of a Flex operator. The additive optimization does not apply when functions are used inside a UDF.

The following Flex stream computes the sum in the ON clause additively, since the `sum()` function is computed additively and the used eventCaches (e0, e1) are declared locally.

```
CREATE INPUT WINDOW In1
    SCHEMA (c1 INTEGER, c2 STRING, c3 INTEGER, summary FLOAT)
    PRIMARY KEY (c1, c2);
CREATE FLEX MyFlex
    IN In1
    OUT OUTPUT WINDOW FlexOut
    SCHEMA (c1 INTEGER, c2 INTEGER, c3 INTEGER, c4 INTEGER)
    PRIMARY KEY (c1, c2)
BEGIN
    declare
        eventCache(In1, coalesce) e0;
        eventCache(In1, coalesce) e1;
    end;
    ON In1 {
     {
        output setOpcode([c1=In1.c1;c2=In1.c2;|
c3=sum(e0.c1);c4=sum(e1.c3);],getOpcode(In1));
     }
    };
END;
```

The following Flex stream is not computed additively, since the `stddev()` function cannot be computed additively.

```
CREATE INPUT WINDOW In1
    SCHEMA (c1 INTEGER, c2 STRING, c3 INTEGER)
    PRIMARY KEY (c1, c2);
CREATE FLEX MyFlex
    IN In1
    OUT OUTPUT WINDOW FlexOut
    SCHEMA (c1 INTEGER, c2 INTEGER, c3 INTEGER, c4 FLOAT)
    PRIMARY KEY (c1, c2)
BEGIN
```

```
    declare
        eventCache(In1, coalesce) e0;
        eventCache(In1, coalesce) e1;
    end;
    ON In1 {
     {
        output setOpcode([c1=In1.c1;c2=In1.c2;|
c3=sum(e0.c1);c4=stddev(e1.c3);],getOpcode(In1));
     }
    };
END;
```

Another restriction is that additive optimizations are disabled when functions are used inside nonlinear statements (IF, WHILE, FOR, and CASE statements). To enable additive optimizations when using a function within a nonlinear statement, assign the result of the function to a variable outside of the statement. Then use the variable inside the nonlinear statement.

> **i** Note
>
> The function used within the nonlinear statement must be from the set of functions eligible for additive optimization.

The following SELECT statement is not optimized for additive optimization since one of the expressions (CASE) in the SELECT list is a nonlinear expression:

```
CREATE OUTPUT WINDOW AggResult
    SCHEMA (id INTEGER, name STRING, counter INTEGER, summary FLOAT)
    PRIMARY KEY DEDUCED
AS
    SELECT BaseInput.intData_1 AS id,
        valueInserted(BaseInput.strData_1) AS name,
        CASE WHEN (count(BaseInput.intDATA_1) < 100) THEN 0 ELSE 1 END AS
counter,
        sum(BaseInput.dblData_1) AS summary
FROM BaseInput
GROUP BY BaseInput.intData_1;
```

4.10 Switch from Decimal to Money Datatype

Using the `money` datatype rather than the `decimal` datatype can improve the performance of a project.

The `money` datatype cannot handle the full range of values that the `decimal` datatype can. Ensure that you do not expect any values outside of the `money` datatype's range before making this change.

4.11 Recompiling Streaming Projects

Recompile existing streaming projects that were compiled with earlier compiler versions to take advantage of the latest enhancements.

Context

Enhancements to the compiler are made constantly to improve the performance of SAP HANA streaming analytics projects.

Procedure

1. Back up the existing CCX file for the project. By default, these files are in the `%`*STREAMING_HOME*`%\bin` folder on Windows machines and the `$`*STREAMING_HOME*`/bin` directory on Linux machines.
2. Open the SAP HANA Streaming Run-Test perspective, and compile the streaming project. Refer to *Compiling a Streaming Project* in the *SAP HANA Streaming Analytics: Developer Guide* for details.

Related Information

Compiling a Streaming Project

4.12 Improving Studio Performance

When you have multiple projects running, studio can slow down. You can improve performance by adjusting preferences in the SAP HANA Streaming Run-Test perspective.

Preference	Location	What to do	Result
Always start previously running run-test views on project start	**Window** > **Preferences** > SAP HANA streaming analytics > **Run Test**	Disable	Prevents previously open views (such as Stream View) from reopening automatically when you connect to a project.
Automatically reconnect all projects when connecting to a server	**Window** > **Preferences** > SAP HANA streaming analytics > **Run Test**	Disable	Prevents projects from automatically connecting when starting studio. Projects must be individually connected to in order to manage them and view activity.

Preference	Location	What to do	Result
Streamviewer pulsed subscribe interval (seconds) **Other pulsed subscribe interval (seconds)**	**Window** > **Preferences** > SAP HANA streaming analytics > **Run Test**	Set to non-zero values	Groups stream updates to be released as often as prescribed, preventing them from occurring continuously.
Filter Meta-data Streams	Server View icon	Enable	Prevents metadata streams from loading with the input and output windows of a project in Server View.

5 Installation Issues

Troubleshoot issues that occur when installing SAP HANA streaming analytics.

To find out about issues fixed in a specific revision, see the SAP Note for that revision on SAP Service Marketplace.

In this section:

Related Information

2659633 - SAP HANA Streaming Analytics 2.0 SP 04 Release Note
2659633 - SAP HANA Streaming Analytics 2.0 SP 04 Release Note

5.1 Uninstallation of SAP HANA Database Fails

Issue: Uninstallation of an SAP HANA database fails due to an inability to remove the host file system. If the SAP HANA database has other SAP HANA capabilities installed, the error message may include references to insufficient permissions or specific files or directories not being found.

An error message similar to the following appears:

```
Uninstalling host <hostname> Terminated
```

This behavior occurs if `.nfsXX` files are detected during the uninstallation process.

Solution:

1. Log on to each worker and standby host in the SAP HANA system as the system administrator (`<sid>adm`).
2. To stop the SAP HANA host service, execute:

   ```
   HDB STOP
   ```
3. Repeat these steps on each of the installed worker and standby hosts.
4. Log on to the SAP HANA database host as the root user.
5. Manually remove the `.nfsXX` files.
   To list the `.nfsXX` files, type:

   ```
   find /hana/shared/<SID>/streaming -name .nfs\*
   ```

   where `<sid>` is the HANA system ID.
   Output example:

   ```
   -rwxr-x--- 1 k9gadm sapsys 4890507 Oct 31 05:14 .nfs000000000724c09200000059
   -rwxr-x--- 1 k9gadm sapsys 1679448 Oct 31 20:49 .nfs000000000724c0450000005a
   ```

   If the `.nfsXX` files report permission errors while manually removing them, shut down the NFS server before removing.
   - To stop the NFS server, execute:

     ```
     /etc/init.d/nfsserver stop
     ```
   - To start the NFS server, execute:

     ```
     /etc/init.d/nfsserver start.
     ```

     > **i** Note
     >
     > Consult your operating system manual or system administrator for assistance.
6. Rerun the SAP HANA platform lifecycle management tool to uninstall the SAP HANA database.

5.2 Unable to Connect to Streaming Server

Issue: After a successful installation of streaming analytics, attempting to connect to the streaming analytics server through studio fails with the following error: `Failed to login server`.

Solution: This issue may be a result of not adding the streaming role or host.

Ensure that all of the installation and setup steps have been completed, specifically focusing on the following areas. Full installation instructions can be found in the *Installing SAP HANA Streaming Analytics* section.

1. Depending on your deployment scenario, add the streaming analytics role or host:
   1. Dedicated deployment, see the following topics:
      - *Add a Streaming Analytics Host Using a Graphical Interface*
      - *Add a Streaming Analytics Host Using a Console Interface*
   2. Same host deployment, see the following topics:

- *Add the Streaming Analytics Role Using a Graphical Interface*
- *Add the Streaming Analytics Role Using a Console Interface*

2. Provision and initialize the streaming analytics. See *Provisioning the Streaming Analytics Service to a Tenant Database* for instructions.
3. Ensure that the hostname and port number you are using are correct. The format for port numbers is `3<instance-number><port-number>`. See *Configuring a Cluster Connection* in the *SAP HANA Streaming Analytics: Developer Guide* for more information. Always enable the SSL checkbox when creating a new server connection.
   The external port is always the internal port number plus one. For example, in single tenant systems, if the internal port is 3xx16 (where xx is the instance number) then the external port is 3xx26. In multiple tenant systems, if the internal port is 3xx40 then the external port is 3xx41.
   To see your assigned internal port number, look at the `streamingserver` service in the landscape view in SAP HANA studio or run the `SELECT PORT FROM M_SERVICES WHERE SERVICE_NAME='<streamingserver>' AND HOST='<host name>'` statement and look at the PORT column in the `M_SERVICES` view.
4. Ensure that the streaming analytics server is running.

Related Information

SAP HANA Streaming Analytics: Developer Guide

5.3 Cannot Initialize the Streaming Service

Issue: When doing an upgrade or a fresh installation of streaming, your attempt to initialize the streaming service fails. This issue may be caused by leftover files or users in your installation.

An error message similar to the following appears:

```
Call to streamingclusterutil returned: 1 There has been an error retrieving the
current cluster configuration
/usr/sap/ETD/HDB00/streaming/cluster/etd/config/current_config.xml (No such file
or directory)
```

Solution:

1. Manually remove the streaming database users and files for each affected tenant database:
   1. Drop the SYS_STREAMING and SYS_STREAMING_ADMIN users using the cascade option.

      ```
      DROP USER SYS_STREAMING CASCADE;
      DROP USER SYS_STREAMING_ADMIN CASCADE;
      ```

   2. Delete these streaming files:
      1. `rm -rf $DIR_INSTANCE/streaming/cluster/<lowercase tenant name>`
      2. `rm -rf /hana/data_streaming/<SID>/cluster/adapters/<lowercase tenant name>`
      3. `rm -rf /hana/data_streaming/<SID>/cluster/projects/<lowercase tenant name>`
      4. `rm -rf /hana/data_streaming/<SID>/<lowercase tenant name>`

2. Once you've added the streaming service to the tenant database, initialize it by running:

```
ALTER SYSTEM INITIALIZE SERVICE 'streamingserver' WITH CREDENTIAL TYPE
'PASSWORD' USING '<desired password>';
```

5.4 Unable to Upgrade After Failed Upgrade Attempt

Issue: After an unsuccessful upgrade attempt, resolving the original issue and attempting to upgrade again fails.

An error message similar to the following appears:

```
SAP HANA streaming
                            analytics: Update to version <version> is pending
(since <timestamp>) at step 'Resume online update'.
```

The hdblcm.log shows the following additional information:

```
INFO: Output line 1: An active installation of this build of SAP HANA streaming
                            analytics <version> is already present in /hana/
shared/<SID>/streaming-<version>, no changes will be made.
```

Solution:

1. Set the BYPASS_STREAMING_VERSION_CHECK environment variable to true: `export BYPASS_STREAMING_VERSION_CHECK=true`
2. Use hdblcm to start the upgrade.

6 Administration Issues

Troubleshoot connection and login issues in SAP HANA streaming analytics.

In this section:

6.1 Adapter Does Not Load Files Through Symbolic Link

Issue: When an adapter tries to access data through a symbolic link, the adapter fails to load it with an `access denied` error.

When you configure an adapter, you specify a relative path for the adapter to use when reading or writing to files. This relative path is specific to the workspace the adapter is in. The sandboxing security feature impacts this, as it prevents adapters and log stores from accessing data outside the specified directory. In streaming analytics, sandboxing is enabled by default, and cannot be disabled.

Solution:

1. Place your files in the correct directory: `$STREAMING_SHARED/<tenant-database-name>/adapters/<workspace-name>`.
   To use a symbolic link, the target of the symbolic link should also be in this directory.
2. If you created the symbolic link while the adapter was already running, you may also run into permissions errors. Make sure that you:
   1. Set file permissions for the symbolic link and for the target.
   2. Restart the java process that reads the directory, and then restart the adapter or project.

6.2 An Unmanaged External Adapter Fails to Connect to a Project

Issue: When starting an external adapter in unmanaged mode without editing its sample XML configuration file, the adapter fails to start.

The external adapter may be unable to connect to the example workspace specified in the sample XML adapter configuration file if the URI specified in the file uses `esp` instead of `esps`. Using `esp` causes the adapter to fail to connect because SSL is enabled by default on SAP HANA streaming analytics and cannot be disabled.

Solution:

1. Open the adapter's XML configuration file in a notepad program and ensure the URI uses `esps`, not `esp`.
2. If using one of the adapter examples provided with your installation, edit the `set_example_env.bat` or `set_example_env.sh` script file to specify:

   ```
   set ADAPTER_EXAMPLE_CLUSTER_NODE_PROTOCOL=esps
   ```

6.3 Cannot Add a Server Connection in Studio

Issue: When working in studio, you cannot add a server connection through the SAP HANA Streaming Run-Test perspective server view.

When adding a server connection, you receive the following error:

```
Invalid Cluster Manager Field, enter a valid Cluster Manager Value.
```

Solution: This error occurs when the value in the **Host Name** field is entered incorrectly.

To fix this issue, enter a valid host in the **Host Name** field, without the `esp` or `esps` prefix. For example, `myhostname`, but not `esps://myhostname`.

6.4 Cannot Connect to the Cluster

Issue: When running a project, you cannot connect to the SAP HANA streaming analytics cluster.

Solution: Check the status of the streaming analytics server, and restart if necessary.

1. Verify that the streaming analytics host is running. See Verify that the Streaming Service is Installed and Running.
2. If the host is not running, start it. See Start the Streaming Service Manually.
3. If the host cannot be started, check the streaming analytics log files in the SAP HANA trace directory for errors.

If you see the error code 700283, there are problems with your streaming analytics license and you may need to get a permanent license key. See *Install a Permanent License* in the *SAP HANA Streaming Analytics: Installation and Update Guide* for detailed instructions.

4. Verify that the SAP HANA index server is running.

> **i** Note
>
> A common cause for issues with starting a streaming analytics host is the SAP HANA index server not running.

Related Information

Verify that the Streaming Service is Installed and Running
Start the Streaming Service Manually
Install a Permanent License

6.5 Cannot Connect to Server

Issue: When trying to run a project, studio returns a connection error.

When running a project, the following error message appears:

```
<Cannot connect to server>
```

This error may be due to an invalid license key.

Solution: View the studio log file.

1. Select **Help > About Studio**.
2. Click **Configuration Details**.
3. Click **Configuration**.
4. Click **View Error Log**.
5. If prompted, select a text editor to view the file.
6. Locate the error. If the log file entry indicates a problem with your license (error code 700283), you may need to get a permanent license key. See *Install a Permanent License* in the *SAP HANA Streaming Analytics: Installation and Update Guide*.

6.6 Error: Invalid Login Credentials

Issue: When you attempt to log in to SAP HANA streaming analytics, an error displays.

```
[error] security : Authentication failure:Invalid login credentials
```

Solution: Verify that the user ID and password are valid and spelled correctly.

6.7 An External Adapter Fails to Start

Issue: Your attempts to start an external adapter fail.

When attempting to run an external adapter an error message similar to the following appears:

```
Failed call to:https://<Streaming-hostname>:61308/RPC2 (Failed to read server's
response: <Streaming-hostname>) java.io.IOException: Failed call to:https://
<Streaming-hostname>:61308/RPC2 (Failed to read server's response: <Streaming-
hostname>)
```

This error is an example of the streaming analytics server not being resolved.

Solution: Use the `ping` command to verify that the hostname of the server to which you are trying to connect can be resolved. If the hostname cannot be resolved:

1. Determine the IP address of the host on which the streaming analytics server is running. Run this command from that machine:

   ```
   nslookup <hostname>
   ```

2. Add the following line to `C:\Windows\System32\drivers\etc\hosts` (Windows) or `/etc/hosts` (UNIX):

   ```
   <ip-address-of-server-hostname>        <Server-hostname>
   ```

6.8 A Studio Project Does Not Run, Reports Login Failure

Issue: You are unable to run a project in a cluster and errors display.

```
Failed to connect to server "esps://<host>:3<instance-number><port-number>".
Reason: "Failed to login server"
```

Studio reports these errors when it has an SSL mismatch with the streaming analytics server. For example, since SSL is enabled on the server by default and cannot be disabled, the mismatch results from the studio connection definition for that server not specifying SSL.

Solution: Correct the connection definition in studio. For details, see Configuring a Cluster Connection.

6.9 A Utility Fails to Connect to the Server

Issue: You are unable to connect to the streaming analytics server using a command-line utility such as `streamingprojectclient` or `streamingsubscribe`.

The command might return this message:

```
Couldn't connect to server, XML-RPC Fault(-504)
```

Utilities affected by this issue include:

- `streamingprojectclient`
- `streamingcnc`
- `streamingconvert`
- `streamingkdbin`
- `streamingkdbout`
- `streamingquery`
- `streamingsubscribe`
- `streamingupload`

Solution: Verify the following:

- You are using the correct host name, port number, and login details in the command.
- The server is up and reachable. You can ping it.
- You are using `-e` correctly. Use the `-e` flag of the utility only if SSL is enabled on the server. For example:

  ```
  > streamingprojectclient -c your_user_name:your_password
  -p <host>:51011/your_workspace/your_project -e
  streamingprojectclient> loglevel 7
  streamingprojectclient> stop
  streamingprojectclient> quit
  ```

  For details on command syntax, see the *SAP HANA Streaming Analytics: Utilities Guide*.

Related Information

SAP HANA Streaming Analytics: Utilities Guide

6.10 A Utility Fails to Connect to a Project

Issue: You cannot connect to a project using a command-line utility such as `streamingprojectclient`.

For example:

```
% streamingprojectclient -c your_user_name:your_password -p <host>:51011/
your_workspace/your_project
```

```
ASAP_loginToCluster( your_user_name ) failed, status = -1
```

Solution: Verify the following:

- The user ID and password are valid and spelled correctly.
- The workspace name and project name are correct.
- If access control is enabled, the user has permissions that allow access to the project.
- The project is running. Use `streamingclusteradmin` to verify and start the project, if necessary.

6.11 Cannot Access Streaming Analytics Pages in SAP HANA Cockpit

Issue: On the system overview in the SAP HANA cockpit, the links to SAP HANA Streaming Analytics pages do not appear.

Solution: Verify that the database user has the CATALOG READ system privilege.

6.12 ODBC Data Source Name Not Found

Issue: When connecting to an ODBC data service, you receive the following error: `[[unixODBC][Driver Manager]Data source name not found, and no default driver specified].`

Solution: Define the data source name (DSN) for the ODBC connection in the `$DIR_INSTANCE/streaming/cluster/<lowercase-db-name>/config/odbc.ini` file.

6.13 Nodes Are Missing from Cockpit and streamingclusteradmin

Issue: On a system with multiple streaming analytics hosts, one or more do not appear in `streamingclusteradmin` or the SAP HANA cockpit.

Solution: On each streaming analytics host, edit the `/etc/hosts` file. Add an entry for the host using the format `<ip-address> <fully-qualified-hostname> <hostname>`. If the `hosts` file has an entry for the host using the format `127.0.0.1 <fully-qualified-hostname> <hostname>`, remove this entry.

6.14 Cannot Start a Project from an External Network

Issue: When working in the SAP HANA studio and connecting to a streaming analytics instance running on an external network (for example, when running SAP HANA streaming analytics on a cloud-hosted instance), you are unable to start a project. Attempting to start a project results in the following two errors:

- Pop-up error:
  `The Cluster has encountered an error starting project <project name> under workspace <workspace name>. Check the project logfile in trace or project working directory for possible cause. Cluster error: [FAILURE:Application current status is not stopped.][CODE:710077]`
- Console output:
  `Failed to call <hostname> (Failed to read server's response: Connection timed out: connect (local) port <port number> to address <IP address>, remote port <port number> to address <IP address> (<hostname>))`

After dismissing the pop-up error message, you see that none of the project elements are displayed under the project in the server view panel. Right-clicking the project will display the option to **Start Project**; however, attempting to start the project again will repeat these errors.

The most likely cause of this combination of error messages is that your streaming analytics project is using ports that are not open for inbound network traffic on the host server, or there is an intermediate firewall.

By default, the ports that your project uses are dynamically assigned at runtime: they change every time you start the project, and are chosen based on what's free at the OS level at the time the project is started. To have a project use specific ports, change the configuration settings of your project in studio.

Solution:

There are two parts to resolving this issue. First, you need to configure your project to use specific ports for the command, SQL, and gateway ports. Secondly, you must configure the security settings for the SDS host and firewall to accept inbound traffic to the streaming server on the chosen ports. Depending on your organization, you may require assistance from your network administrator.

1. From the SAP HANA Streaming Development perspective, in the Project Explorer window, open your project's configuration file (.crr).
2. Open the advanced tab and select **Project Deployment**.
3. Next to **Command Port**, **SQL Port**, and **Gateway Port**, there are suggested port numbers. You can either leave the suggested numbers, or enter your own preferred numbers. Check the boxes next to each setting to make them active, and save the file.
4. Redeploy your project for the new settings to update. Next time you start the project, it will use the specified ports. Use these specific port numbers to configure your network firewall and gain access to your project remotely.

6.15 Error: Could Not Allocate Controller

Issue: When you start a streaming analytics project, the following error displays: `Could not allocate Controller`.

Solution: This error occurs when SAP HANA and streaming analytics are uninstalled and reinstalled with the same SID without first deleting the `/hana/data_streaming/<SID>` folder.

1. Uninstall streaming analytics.
2. Delete the `/hana/data_streaming/<SID>` folder.
3. Reinstall streaming analytics.

6.16 Cannot Start Up Streaming Server

Issue: The streaming server does not start up.

Solution 1: If you have edited the node name of an existing streaming analytics host, the node name in `streamingserver.ini` does not match the name in the cluster configuration file. To solve this, change the node name back to its default value.

To check the correct node name, run the following command as the `<sid>`adm user:

```
hdbstreamingserver -u --config $STREAMING_HOME/../cluster/<db-name>/config/
cluster.cfg --show
```

The node name is in the Nodes section of the output:

```
<Cluster>
    <Nodes>
        <Node enabled="true" name="<node_name>">
```

Set the node name in `streamingserver.ini` to this value and restart all running nodes for the changes to take effect.

Solution 2: This issue may also be caused by an invalid SAP HANA license. Check the streaming server log file. In the file, you see the following messages:

```
Caused by: com.sap.db.jdbc.exceptions.JDBCDriverException: SAP DBTech JDBC:
[591]: internal error: only commands for license handling are allowed in current
state
...
FATAL - CODE_700418 | Could not load config from database
FATAL - CODE_700412 | Factory of new node failed
```

If you see these messages, there is a problem with your SAP HANA license and you may need to get a permanent license key. See Install a Permanent License for detailed steps.

Related Information

Edit the Streaming Node Name
Check the Current License Key
Managing SAP HANA Licenses

6.17 Cannot Stop Project

Issue: When your project is hanging and you attempt to stop it, the project status remains at "running", and the project keeps running.

Solution: Force the project to stop, which immediately shuts down all project and related processes. You can do this in one of the following ways:

- Using the `streamingclusteradmin` utility, run the `--stop_project` command with the `timeout` option set to -1:

```
$STREAMING_HOME/bin/streamingclusteradmin --uri=esps://<host>:<port> --
username=<username> --password=<password> --stop_project --workspace-
name=<workspace> --project-name=<project> --timeout -1
```

- Using SAP Web IDE for SAP HANA, open the streaming analytics runtime tool. Expand the workspace on which the project is running, then expand the **Projects** folder, right-click on the frozen project and select **Force Stop Project**.
- Using the SAP HANA cockpit, on the SAP HANA system overview, under **Streaming Analytics**, click **Manage Streaming Projects**. From the projects list, select the frozen project and click **Force Stop**.
- Using the SAP HANA studio, open the SAP HANA Streaming Run-Test perspective. In the stream view, right-click on the frozen project and select **Force Stop Project**.

6.18 Cannot Open a WebSocket Connection

Issue: You cannot open a WebSocket connection due to self-signed certificates.

Solution: The Streaming Web Service and the Web Services Provider come installed by default with self-signed certificates. However, some browsers require user permission to accept these certificates.

To work around this issue, consult the specific browser documentation and configure your browser to accept self-signed certificates. You can also:

- (Streaming Web Service) Replace the SSL certificate file with a certificate file that has been signed by a certificate authority (CA). See Configuring SSL for the Streaming Web Service for more information.
- (Web Services Provider) Modify the keystore to contain a certificate which has been signed by a CA. See Configuring SSL for the Web Services Provider for more information.

6.19 Web Service Provider Connection Fails

Issue: Connection to the Web Service Provider fails.

If the client origin changes and no longer matches the value in the Web Service Provider's `allowOrigin` property, the connection to the Web Services Provider fails.

Solution: Configure the client to use a static origin and set `allowOrigin` to this value.

7 Development Issues

Troubleshoot issues with developing projects and using the streaming analytics plugin for SAP HANA studio.

In this section:

Issue: When using multiple external managed adapters in a project, you cannot implicitly guarantee that they start in the desired order.

7.1 Cannot Connect to the Cluster

Issue: When running a project, you cannot connect to the SAP HANA streaming analytics cluster.

Solution: Check the status of the streaming analytics server, and restart if necessary.

1. Verify that the streaming analytics host is running. See Verify that the Streaming Service is Installed and Running.
2. If the host is not running, start it. See Start the Streaming Service Manually.
3. If the host cannot be started, check the streaming analytics log files in the SAP HANA trace directory for errors.
   If you see the error code 700283, there are problems with your streaming analytics license and you may need to get a permanent license key. See *Install a Permanent License* in the *SAP HANA Streaming Analytics: Installation and Update Guide* for detailed instructions.
4. Verify that the SAP HANA index server is running.

> **i** Note
>
> A common cause for issues with starting a streaming analytics host is the SAP HANA index server not running.

Related Information

Verify that the Streaming Service is Installed and Running
Start the Streaming Service Manually
Install a Permanent License

7.2 Cannot Start an Adapter in Cluster-Managed Mode

Issue: Your attempts to start an adapter in cluster-managed mode fail.

Symptom: The following error message displays in the project output file:

```
ERROR [main] (PortAllocator.allocateFromDataCenter) Fail to get global lock.
```

If the adapter fails to start, it may be due to a global locking issue, which can happen when streaming analytics is installed on a Network File System (NFS) shared disk.

Solution: In the `$STREAMING_HOME/adapters/framework/bin/` folder, edit the start.bat (Windows) or start.sh (Unix) file to set **`STREAMING_ADAPTER_FRAMEWORK_DATA_DIR`** to a path on your local machine.

7.3 Schema Discovery Fails

Issue: When executing discovery for an element in a streaming project, discovery fails.

Discovery uses the Default Server URL preference to retrieve the list of available data services. The default server URL is esp://localhost:9786. Because there is no localhost streaming analytics server, no data services are located.

Solution: Change the Default Server URL preference to an available streaming analytics server.

1. In the SAP HANA Streaming Run-Test perspective, select **Window** > **Preferences**.
2. In the Preferences dialog, select **SAP HANA streaming analytics**.
3. In the **Default Server URL** field, click **Change** and select a server from the Select Default Server URL dialog. Click **OK**.
4. In the Preferences dialog, click **Apply**, then **OK**.

7.4 Changing Date Format for Playback

Issue: Unable to load or playback files containing dates in formats other than UTC.

The studio playback tool assumes that all dates and times are in UTC format.

Solution: Set a date mask to change the order of date and time values.

1. Open the SAP HANA Streaming Run-Test perspective.
2. In the Playback view, enter a date mask in the **XML/CSV datemask** field using the following format: `%Y-%m-%dT%H:%M:%S`.
   Change the value order as necessary.

You cannot change delimiters from within studio. To learn how to specify a different delimiter from the command line, see *Server Executables* in the *SAP HANA Streaming Analytics: Utilities Guide*.

Related Information

SAP HANA Streaming Analytics: Utilities Guide
Playback
Server Executables

7.5 An External Adapter Fails to Start

Issue: Your attempts to start an external adapter fail.

When attempting to run an external adapter an error message similar to the following appears:

```
Failed call to:https://<Streaming-hostname>:61308/RPC2 (Failed to read server's
response: <Streaming-hostname>) java.io.IOException: Failed call to:https://
<Streaming-hostname>:61308/RPC2 (Failed to read server's response: <Streaming-
hostname>)
```

This error is an example of the streaming analytics server not being resolved.

Solution: Use the `ping` command to verify that the hostname of the server to which you are trying to connect can be resolved. If the hostname cannot be resolved:

1. Determine the IP address of the host on which the streaming analytics server is running. Run this command from that machine:

   ```
   nslookup <hostname>
   ```

2. Add the following line to `C:\Windows\System32\drivers\etc\hosts` (Windows) or `/etc/hosts` (UNIX):

   ```
   <ip-address-of-server-hostname>        <Server-hostname>
   ```

7.6 A Kafka Adapter Fails to Start

Issue: A Kafka adapter fails to start when trying to use it in a project.

If your Kafka adapter is not working, verify that you have installed the Kafka client and that all required files are in their proper location.

Before using any Kafka adapter, copy over certain `.jar` files to your streaming analytics installation.

1. For all Kafka adapters:
   - Navigate to Apache Kafka download mirrors .
   - Download and extract the Kafka binary, and then copy the `kafka-clients-0.9.0.0.jar` client file to `STREAMING_CUSTOM_ADAPTERS_HOME/libj`.
2. For the Kafka Avro Record Input and Output adapters only:

- Navigate to Apache Avro download mirrors , select one of the mirror sites, and navigate to the `avro-1.7.7/java/` directory.
- Copy the `avro-1.7.7.jar` file to `STREAMING_CUSTOM_ADAPTERS_HOME/libj`.
- Navigate to Jackson core download index .
- Copy the `jackson-core-asl-1.9.13.jar` and `jackson-mapper-asl-1.9.13.jar` files to `STREAMING_CUSTOM_ADAPTERS_HOME/libj`.

7.7 Playback is Too Fast or Slow

Issue: While using the playback tool in the SAP HANA Streaming Run-Test perspective, data plays back too fast or too slow.

By default, studio runs projects in Full Rate mode. This playback mode is dependent on external factors, such as the computer running .

Solution: Change the playback mode and adjust the playback rate.

1. Open the SAP HANA Streaming Run-Test perspective.
2. In the Playback view, select the **rec/ms** playback mode.
3. Set your preferred speed in the **rec/ms** field. For example, set your speed to 0.01 to run the project at a speed of 0.01 records per millisecond.
4. Run your project. If necessary, adjust the speed setting using the **At timestamp rate** slider. Using the slider allows you to change playback speed while the project is running, since the **rec/ms** setting can only be directly changed while a project is stopped.

7.8 A Project Triggers Java Out-of-Memory Errors

Issue: When a project runs, you receive out-of-memory errors from the Java virtual machine.

Solution: Modify the project configuration (.ccr) file to increase the heap size for the project's Java virtual machine.

7.9 Published Data Lost When Project Fails

Issue: A subscriber receiving data through a stream does not receive all the data sent by the publisher before the streaming analytics node shuts down unexpectedly.

Solution: Enable guaranteed delivery on the stream, or replace it with a guaranteed delivery-enabled window.

Guaranteed delivery (GD) uses log stores to ensure that a GD subscriber registered with a GD stream or window receives all the data processed by that stream or window even if the client is not connected when the

data is produced. GD is supported on streams and windows and each GD stream or window requires a GD log store.

> **i Note**
>
> GD-enabled windows are not recommended on a project configured for hot standby. The shared disk requirements for GD log stores are incompatible with the continuous synchronization that enables a primary instance to fail over quickly to its secondary instance.

See the *SAP HANA Streaming Analytics: Developer Guide* for information on guaranteed delivery.

Related Information

Guaranteed Delivery

7.10 Stream View Causes Project to Run Slower

Issue: A project runs slower when streams or windows are open in the Stream View.

As studio tries to keep up with the data flowing through the server, it blocks incoming data and slows down the project.

Solution: Enable lossy subscription. This option allows the studio subscription to lose data if it is unable to keep up with the data flowing through the server.

1. Go to **Window** > **Preferences** > **SAP HANA streaming analytics** > **Run Test** .
2. Select **Streamviewer lossy subscription**. Click **OK** to save.

7.11 Stream View Displays Partial Data

Issue: In the SAP HANA Streaming Run-Test perspective, the Stream View does not show all rows of a running project.

Solution: Increase the number of visible rows in the Stream View preferences.

> **i Note**
>
> Increasing the number of rows also increases memory usage in studio.

1. Open the SAP HANA Streaming Run-Test perspective.
2. In the top right corner of the Stream View, select **Set StreamViewer Number of Rows Displayed**  .
3. Enter a new number of rows, then click **OK** to save.

4. (Optional) If your project has stopped running, rerun the project to see the new number of rows.

For more information on Stream View preferences, see the *Viewing a Stream* topic in the *SAP HANA Streaming Analytics: Developer Guide*.

If Stream View does not show any data, the data may have already been processed. See Stream View Displays No Data [page 80] for more information.

Related Information

Viewing a Stream

7.12 Stream View Displays No Data

Issue: When running a project that uses one or more file input adapters, Stream View does not show any data.

File input adapters read data into projects at very high speeds. Streams are stateless and do not store any data, so by the time a user opens a stream in the viewer, all the input data has already been processed. Since the Stream View can only show new events since the time it was opened, there is nothing to see.

Solution: Manually load the data using the File Upload tool.

1. In the SAP HANA Streaming Development perspective, remove the ATTACH ADAPTER statement from the CCL, or delete the adapter using the visual editor.
2. Compile and run the project.
3. In the SAP HANA Streaming Run-Test perspective, open the necessary streams in Stream View.
4. Open the File Upload tool to manually load the data. For more information, see the *Uploading Data to the SAP HANA Streaming Analytics Server* topic in the *SAP HANA Streaming Analytics: Developer Guide*.

If Stream View shows some data but the rest is missing, adjust your preferences to increase the number of visible rows. See Stream View Displays Partial Data [page 79] for more information.

Related Information

Uploading Data to the SAP HANA Streaming Analytics Server

7.13 Studio Crashes and You are Unable to Restart It

Issue: The studio workspace may have become corrupt.

Solution: Migrate your projects to a new workspace.

1. Rename the corrupt workspace by appending the string `_OLD` to its name.
2. Shut down and restart studio. Upon restarting, studio creates a new workspace folder.
3. In the SAP HANA Streaming Development perspective, import your projects into the new workspace. See *Creating or Modifying a Streaming Project in SAP HANA Studio* in the *SAP HANA Streaming Analytics: Developer Guide* for detailed steps.

Related Information

Creating or Modifying a Streaming Project in SAP HANA Studio

7.14 A Utility Fails to Connect to a Project

Issue: You cannot connect to a project using a command-line utility such as `streamingprojectclient`.

For example:

```
% streamingprojectclient -c your_user_name:your_password -p <host>:51011/
your_workspace/your_project
ASAP_loginToCluster( your_user_name ) failed, status = -1
```

Solution: Verify the following:

- The user ID and password are valid and spelled correctly.
- The workspace name and project name are correct.
- If access control is enabled, the user has permissions that allow access to the project.
- The project is running. Use `streamingclusteradmin` to verify and start the project, if necessary.

7.15 External Managed Adapters Do Not Start in Desired Order

Issue: When using multiple external managed adapters in a project, you cannot implicitly guarantee that they start in the desired order.

Solution: Enforce a specific ordering of external managed adapters:

1. In your CCL project, use the ADAPTER START statement with the NOSTART clause to prevent the adapters from starting with the project. For more information and an example of usage, see the *ADAPTER START Statement* topic in the *SAP HANA Streaming Analytics: CCL Reference*.
2. Use the `streamingprojectclient` utility to start the adapters in your desired order. You can use a script to automate the command calls, and add short delays as necessary to ensure that all adapters are called and started in your desired order.

Related Information

ADAPTER START Statement

7.16 Error Compiling CCL in PowerDesigner

Issue: You receive the following error when importing CCL into a Streaming Schema model in PowerDesigner:

```
There was an error in compiling the given CCL. Please check the log file.
```

This issue occurs on Windows 2008 R2 if PowerDesigner is installed in the system program folder. The default permission for this folder is ReadOnly, which prevents PowerDesigner from writing to this location.

Solution: Add the user Everyone to the PowerDesigner installation folder and grant this user read and write permissions.

7.17 User Cannot Perform Tasks Despite Being Granted Permissions

Issue: Despite having been granted permission to perform a task, a user receives an error message stating that they do not have the required permission for the operation.

Solution: You may have granted permission to a user that does not exist in the SAP HANA system. Verify that the username for the user to whom you are granting a permission matches their SAP HANA username:

- If you are using the SAP HANA cockpit, verify that the username under the **Users** tab in the Manage Streaming Permissions page matches the SAP HANA username.

  > **i** Note
  >
  > When you add a user to the Manage Streaming Permissions page, you are not creating an SAP HANA user. Users listed under the **Users** column are only users that have been granted at least one streaming analytics permission or role.

- If you are using the `streamingclusteradmin` utility, use the `get users` and `get perms` commands to verify that the streaming analytics username matches the SAP HANA login credentials.
- If you are using the SAP HANA studio, you are not a restricted user, and you have the necessary admin permissions, you can reference the list of all current SAP HANA users by doing the following:
  1. In the **Systems** view of the **SAP HANA Administrator Console** perspective, log on to the necessary system as an admin user.
  2. Expand the **Security** folder.
  3. Expand the **Users** folder to see a comprehensive list of all SAP HANA users on the system. Use this list as a reference to ensure that the username you are granting permissions to on the SAP HANA cockpit Manage Streaming Permissions page matches the username you see in this list.

For information on how to set up and authenticate SAP HANA users, see the *Managing SAP HANA Users* section in the *SAP HANA Administration Guide*.

Related Information

SAP HANA Administration Guide

7.18 CCLScript Operations on Records Fail

Issue: When you assign or get fields in a record, the operation fails.

Solution: Initialize the record variable before assigning or getting individual fields.

7.19 Cannot View Input and Output Adapters in the Palette

Issue: The input adapters and output adapters folders appear empty in the Studio palette.

Solution: Install the streaming analytics plugin by starting SAP HANA studio with "Run as administrator" permissions on Windows.

1. Start SAP HANA studio with "**Run as administrator**" permissions.
2. Uninstall the streaming analytics plugin.
3. Restart SAP HANA studio with "**Run as administrator**" permissions to complete the uninstallation.
4. Reinstall the streaming analytics plugin while studio is running with "**Run as administrator**" permissions.

7.20 SAP IQ Output Adapter Blocked from Writing to a Table

Issue: You receive this error message while using the SAP IQ output adapter to load files into an SAP IQ table: `User <username> has the row in <table_name> locked.`

This error occurs if, for example, your SAP IQ host crashes as the adapter is loading files into the table. When this happens, the table may still be locked by the primary host when the adapter connects to the secondary host and attempts to load the file. The error can also occur if you attempt to write to a table that is locked by another user.

Solution: To work around this, use the LOAD TABLE statement provided in the server log to manually reload the file.

7.21 Stream View Shows Date Values in UTC or Local Time

Issue: Stream View interprets date values as UTC or the local time zone, and you want to change this behavior.

Solution:

1. In studio, choose **Window** > **Preferences**, then expand > **Run Test** and select **Manual Input Settings**.
2. Check the **Interpret Date values in Manual Input and Stream Viewer as UTC** option and select **Apply**.

7.22 Studio is Slow When Establishing a Server Connection

Issue: You are working in studio to establish an initial connection to a server that hosts projects, and the loading process is very slow.

When studio is not located on the same network as the server, connecting to the server may take longer. Accessing multiple projects at once on the server will further stall loading times. If you have enabled the Run-Test preference **Automatically reconnect to all projects when connecting to a server**, all of your projects will automatically connect when you access a server, causing this lag.

Solution: Change your SAP HANA Streaming Run-Test preferences so that projects do not automatically reconnect.

1. Go to **Window** > **Preferences** > **SAP HANA streaming analytics** > **Run Test**.
2. Uncheck **Automatically reconnect to all projects when connecting to a server**. Click **OK** to save.
3. From the SAP HANA Streaming Run-Test perspective, right-click an individual project in the Server View to connect to or disconnect from that project.

7.23 Project Binding Fails to Connect

Issue: Adding a binding between projects results in an error where the binding cannot connect to one of the projects.

> **i** Note
>
> This functionality is not supported in streaming lite.

Prerequisites: Ensure that your hostname can be resolved to an IP address by DNS. If not, edit the /etc/hosts file to add the IP address and hostname for the streaming host to the file using the following format:

```
<ip-address> <streaming-hostname>
```

Solution: Configure streaming analytics to connect to the server without a proxy. Normally, `no_proxy` uses domain names to allow host connections without a proxy. However, streaming analytics connects to the server without the use of a domain name, which means that the connection would use a proxy. To allow streaming analytics to connect without a proxy, add the server hostname without the domain name to `no_proxy`.

1. In your Linux environment, run `set|grep proxy`. This shows you what is currently on the `no_proxy` list.
2. Add your hostname to `no_proxy`. In your Linux .bashrc configuration file, add `export no_proxy=<hostname>,$no_proxy`.
3. Restart your projects.

7.24 A File Input Adapter Fails to Read Records From a File

Issue: When using an adapter that reads input from a file, the adapter fails to successfully read and send records into the project.

Solution 1: Change the Input Stream to an Input Window

One way to ensure a File Input adapter is successfully reading records from an input file is to change the stream it is feeding to an input window.

By using a window instead of a stream, you can view the records inside the window after they have been added, and ensure that they all appear correctly.

1. Delete the input stream element.
2. Create a new input window using the studio visual editor or using the CREATE WINDOW statement.
3. Set a retention policy for the input window so that it keeps all rows. See *Specifying a Retention Policy* in the *SAP HANA Streaming Analytics: Developer Guide* for more information.
4. Attach the input adapter to the input window.
5. Compile and run the project.
6. Set all of the column types in the input data file to `string`, to facilitate reading and avoid parsing errors.
7. In the SAP HANA Streaming Run-Test perspective, in the server view, expand your project to show the list of windows and streams in the project.
8. Right-click the input window and select **Show in** > **StreamViewer**.

Viewing a stream is possible, but only after a project has started running. The records can also finish flowing through the stream in the time it takes to open the StreamViewer. Using an input window that retains all records as they are added makes troubleshooting file reading problems easier.

If the adapter correctly reads the input from the file, change the window back into a stream to prevent retention policy and out of memory errors, or configure an appropriate retention policy. See *Retention Policies* in the *SAP HANA Streaming Analytics: Developer Guide* for more information.

Solution 2: Look in the .out Log File for Adapter Information and Errors

If the adapter still does not correctly read records from an input file, look in the `.out` file to find the full path to the directory where the adapter expects files to appear. See *Locating Log and Trace Files for Troubleshooting Adapters* in the *SAP HANA Streaming Analytics: Adapters Guide* for more information.

Correct the filepath for the adapter in the adapter's properties in studio, or in the adapter's `.xml` file. See the appropriate configuration topic for your adapter in the *SAP HANA Streaming Analytics: Adapters Guide* for more information.

Related Information

Managing an Input Stream or Window
Adding an Adapter to a Project
Specifying a Retention Policy
Retention Policies

7.25 SAP HANA Streaming Run-Test Runs Stale or Outdated Code

Issue: When using the SAP HANA Streaming Run-Test perspective to test a project, studio runs a stale or outdated version.

Solution:

1. If the project is currently running, stop it.
2. Recompile the project in the SAP HANA Streaming Development perspective.
3. In the SAP HANA Streaming Run-Test perspective, in the server view, right-click your project and select **Remove Project**.
4. Right-click your desired workspace and select **Load Project(s) into Workspace** to re-add your project's `.ccx` file.
5. Right-click the project and select **Start Project**.

7.26 Project Has No Data Flow

Issue: You have set up your project and successfully performed schema discovery for your input adapter. You switch to SAP HANA Streaming Run-Test Perspective and see no data flow.

Solution: In order for schema discovery to successfully read your adapter files, the `dir` property of the adapter must be set to the absolute path of `STREAMING_SHARED/<tenant-db-name>/adapters`, and the files to be read must be located in that directory. However, the project will not run if the `dir` property has an absolute path, so you must change the value to a relative path before starting the project.

1. Ensure that the files for schema discovery are located in `/hana/data_streaming/<SID>/<tenant-db-name>/adapters`. We recommend grouping adapter files into subdirectories for each project.
2. Set the `dir` property on your adapter to the absolute path from the previous step.
3. Run schema discovery with your adapter.

4. Save your project.
5. Edit the `dir` property and change it from the previous absolute path to a relative path (relative to the streaming sandbox directory).
6. Save and start your project.

7.27 Workspace is Missing from the Streaming Runtime Tool

Issue: The workspace pane of the streaming runtime tool is open, but no workspace appears, or a workspace that you need is missing.

Each time you build and deploy a new streaming module from SAP Web IDE, the streaming runtime tool creates a new workspace. If you have not yet built a streaming module, then the workspace for that module does not yet exist.

If you have a streaming project that needs a data service or a machine learning model, you need to first build the module with the unfinished project to generate a workspace, where you can then create data services or PAL models. Make sure that your streaming project contains some CCL elements, as a completely empty streaming project will not build. Alternatively, you can also manually add an existing custom workspace to the tool from any streaming server, provided that you have credentials with the correct streaming permissions.

Solution: Build a streaming module to create a new workspace:

1. Log into the SAP Web IDE.
2. In the **Workspace** pane, right-click the streaming analytics module, and select **Build** to compile and run all projects in the module.
3. Switch over to the streaming runtime tool to see your project in a newly created workspace.

7.28 Cannot Connect to Externally Hosted Project

Issue: In studio, when trying to connect to a project on a different host the connection fails with the error `Fail to call to: <hostname> (Failed to read server's response: <hostname without domain>)`.

The streaming analytics installation truncates its own hostname, removing the domain name. Therefore, when other hosts attempt to connect, there is no domain name to connect with.

Solution:

1. Add the hostname of the streaming host to `no_proxy`. From the console, execute the following:

   ```
   export no_proxy=<hostname>,$no_proxy
   ```

2. As the root user (Linux) or as an administrator (Windows), add the hostname and IP of the streaming host to the /etc/hosts file. Use the following format:

   ```
   <IP address> <full hostname> <hostname without domain>
   ```

   Find the file in Windows at %SystemRoot%\System32\drivers\etc\hosts.

7.29 GDMode or EnableGdMode Property Triggers Illegal Parameter Error

Issue: You have upgraded streaming analytics to 2.0 SP 02 or later, and a project using the `GDMode` or the `EnableGdMode` property is now causing `Illegal Parameter` errors.

Solution: The property IDs for these properties changed as of 2.0 SP 02, which means you'll need to update your project CCL file (managed mode) or adapter XML configuration file (unmanaged mode) with the new IDs. See the table below for the updated property IDs.

1. Stop your project if it isn't already stopped.
2. Open the project CCL file or adapter XML configuration file and update the property IDs as needed.
3. Save, compile, and run your project.

Adapter	Previous Property ID	New Property ID in Managed Mode	New Property ID in Unmanaged Mode
Web Services (SOAP) Output	`GDMode`	N/A	`EnableGdMode`
EspPublisher Input Module			
File/Hadoop CSV Input		`enableGdMode`	
File/Hadoop Event XML Input			
File/Hadoop JSON Input			
JMS CSV Input			
JMS Event XML Input			
JMS Map Input			
JMS Object Input			
Kafka Avro Input	`EnableGdMode` (managed mode) `GDMode` (unmanaged mode)		
Kafka CSV Input			
Kafka Event XML Input			
Kafka JSON Input			
Kafka String Input			

7.30 Stream or Window Does Not Display All Rows

Issue: A stream or window contains a large number of rows, but cannot display any rows over a certain threshold.

If you have too many rows, you may see the following warning message:

```
Maximum number of display items is 100, container contains 120. Increase the
maximum number through studio properties.
```

Solution:

1. Go to **Window** > **Preferences** > **SAP HANA streaming analytics** > **CCL Visual Editor** .
2. Increase the number in the **Maximum number of shape compartment items to display per diagram shape** field.

7.31 Connection Issue Due to SSL Server Certification Validation Error

Issue: When attempting to start the streaming server, it fails to start up.

The following message displays in the streaming server log:

```
ERROR - CODE_710342 | Failed to connect to the database due to SSL server
certificate validation error.
```

Solution:

1. Verify the SAP HANA SSL certificate:
   1. In SAP HANA, under **Database Administration**, select **Manage system configuration**.
   2. In the drop-down menu for **Configuration File**, select **global.ini**, and in the drop-down menu for **Section**, select **communication**.
   3. Note the value of the `sslkeystore` parameter.
   4. As the `<sid>`adm user:
      1. Run `sapgenpse get_my_name -p <hana keystore>`.
      2. Note the certificate issuer. If the certificate is self-signed, you may need to add it to the truststore used by the cluster. Otherwise, you may need to import the root and intermediate CA certificates for this certificate.
      3. Verify the certificate validity. Is the `NotBefore` time after the current time? Is the `NotAfter` time before the current time? If so, you may need to generate an updated certificateand import it into this keystore.
      4. Verify that the system date and time is correct. This may cause the certificate validity check to fail.
2. As the `<sid>`adm user, verify the streaming cluster JDBC truststore:
   1. Open the `$DIR_INSTANCE/streaming/cluster/<lower-case sid>/config/cluster.cfg` file.
   2. Note the value of `jdbc-hana-truststore`.
   3. Run `$STREAMING_HOME/lib/jre/bin/keytool -list -rfc -keystore <cluster truststore path> -storepass <cluster truststore password>`

4. If the server certificate is self-signed, export the SAP HANA SSL certificate and import it into the Streaming cluster JDBC truststore.

```
sapgenpse export_own_cert -p <hana keystore> -b -o server.crt
keytool -importcert -trustcacerts -keystore <cluster truststore path> -
storepass <cluster truststore password> -file server.crt -noprompt
```

5. Otherwise, import the root and intermediate certificates for the SAP HANA SSL certificate into the streaming cluster JDBC truststore.

```
keytool -importcert -trustcacerts -keystore <cluster truststore path> -
storepass <cluster truststore password> -file root.crt -noprompt
keytool -importcert -keystore <cluster truststore path> -storepass
<cluster truststore password> -file intermediate.crt -noprompt
keytool -importcert -keystore <cluster truststore path> -storepass
<cluster truststore password> -file server.crt -noprompt
```

7.32 Connection Issue Due to SSL Keystore Access Error

Issue: When attempting to start the streaming server, it fails to start up.

The following message displays in the streaming server log:

```
ERROR - CODE_710336 | Failed to connect to the database due to error with
configured keystore.
```

Solution:

1. As the <sid>adm user, verify the configuration of the streaming cluster JDBC keystore:
   1. Open the `$DIR_INSTANCE/streaming/cluster/<lowercase_sid>/config/cluster.cfg` file.
   2. Verify if "`jdbc-hana-keystore`" is pointing to the correct filepath.
   3. Verify if "`jdbc-hana-keystore-type`" is present and set it to JKS.
   4. Verify if "`jdbc-hana-keystore-password`" is present, and whether the password is protected and correct.
   5. Verify if "`jdbc-hana-keystore-password-is-encrypted`" is present. If the keystore password is encrypted, re-encrypt the correct password using `streamingencrypt` and compare it to the value of the `jdbc-hana-keystore-password` setting.
2. Use the Java keytool to load the keystore and verify whether the file is uncorrupted and contains a single private key entry. Also, verify that the public key algorithm is RSA. If the keystore cannot be loaded by the keytool or the public key algorithm is not RSA, it may need to be regenerated.

   ```
   keytool -list -keystore mykeystore.jks -storepass password -v
   ```

3. Verify whether SAP HANA JDBC SSL client certificate validation is enabled:
   1. In SAP HANA, under **Database Administration**, select **Manage system configuration**.
   2. In the drop-down menu for **Configuration File**, select **global.ini**, and in the drop-down menu for **Section**, select **communication**.
   3. Verify if the value for `ssl` is either **on** or **systempki**.
   4. Note the values of `sslkeystore`, `ssltruststore`, and `sslvalidatecertificate`.
4. If `sslvalidatecertificate` is true:

1. Verify whether the SAP HANA `sslkeystore` keypair matches the key in the keystore:

   ```
   sapgenpse get_my_name -p <hana keystore>
   ```

2. Verify whether the SAP HANA `ssltruststore` contains the cluster keystore certificate:

   ```
   sapgenpse maintain_pk -p <hana keystore> -l
   ```

3. You can import keystore certificates into the SAP HANA truststore using the following commands:

   ```
   keytool -exportcert -keystore <cluster keystore> -alias <keypair alias> -
   keypass <keypair password> -storepass <cluster keystore password> -file
   export.crt
   sapgenpse maintain_pk -p <hana truststore> -a export.crt
   ```

7.33 Connection Issue Due to SSL Truststore Access Error

Issue: When attempting to start the streaming server, it fails to start up.

The following message displays in the streaming server log:

```
ERROR - CODE_710339 | Failed to connect to the database due to error with
configured truststore.
```

Solution:

1. As the `<sid>`adm user, verify the configuration of the streaming cluster JDBC truststore:
   1. Open the `$DIR_INSTANCE/streaming/cluster/<lower-case sid>/config/cluster.cfg` file.
   2. Verify if `jdbc-hana-truststore` is pointing to the correct filepath.
   3. Verify if `jdbc-hana-truststore-type` is present and set it to JKS.
   4. Verify if `jdbc-hana-truststore-password` is present, and whether the password is protected and correct.
   5. Verify if "`jdbc-hana-truststore-password-is-encrypted`" is present. If the keystore password is encrypted, re-encrypt the correct password using `streamingencrypt` and compare it to the value of the `jdbc-hana-keystore-password` setting.
2. Use the Java keytool to load the truststore and verify whether the file is uncorrupted and contains the expected trusted certificate entries. If the keystore cannot be loaded by the keytool or the public key algorithm is not RSA, it may need to be regenerated.

   ```
   keytool -list -keystore mytruststore.jks -storepass password -v
   ```

7.34 Connection Issue Due to SSL Errors

Issue: When attempting to start the streaming server, it fails to start up.

The following message displays in the streaming server log:

```
ERROR – CODE_710333 | Failed to connect to HANA due to an SSL error.
```

Solution: Check that SSL is enabled for SAP HANA JDBC connections.

1. In SAP HANA, under **Database Administration**, select **Manage system configuration**.
2. In the drop-down menu for **Configuration File**, select **global.ini**, and in the drop-down menu for **Section**, select **communication**.
3. Verify that the value of the `ssl` parameter is either **on** or **systempki**.
   - To configure a custom keystore/truststore for SSL, set the value to **on** and configure the `sslkeystore`, `ssltruststore`, and `sslvalidatecertificate` parameters accordingly.
   - To configure SAP HANA to use the system PKI for SSL, set the value to **systempki**.

7.35 Streaming Web Service Does Not Support NULL Characters in JSON

Issue: When JSON strings contain the NULL character, you get errors when publishing or subscribing using the Streaming Web Service.

When sending JSON input to the Streaming Web Service which contains a JSON string with the NULL character, the Streaming Web Service returns error 400 Bad Request.

When subscribing to a stream using the Streaming Web Service, strings which contain unicode NULL characters are truncated.

Solution: If the data contains the NULL character, then use the Web Services Provider for publishing and subscribing.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

THE BEST RUN SAP