SAP Event Stream Processor: Getting Started Guide
# Table of Contents

1  SAP Event Stream Processor Overview. ................................................................. 4
1.1  Key Terms and Concepts. ...................................................................................... 5
1.1.1  Events. .............................................................................................................. 5
1.1.2  Operation Codes. ............................................................................................. 6
1.1.3  Streams and Windows. .................................................................................... 7
1.1.4  Streams, Windows, Adapters, and Continuous Queries in Projects. ................. 8
1.1.5  Data-Flow Programming. ................................................................................ 8
1.1.6  Getting Results from a Project. ...................................................................... 10
1.1.7  SAP HANA. ..................................................................................................... 11
1.2  User Roles and Capabilities. ................................................................................. 11
1.3  Beyond the Basics. ............................................................................................... 13

2  Getting Started with a Project. .................................................................................. 14
2.1  Launching Studio. ................................................................................................. 14
2.2  Exploring Studio Perspectives and Views. ............................................................ 15
2.3  The Authoring Perspective. .................................................................................. 17
2.4  The Run-Test Perspective. ................................................................................... 19
2.5  Diagrams. ............................................................................................................. 21
2.6  Learning through Sample Projects. ..................................................................... 22
2.6.1  How to load a Sample Project. ........................................................................ 24
2.7  Project Execution and Testing. ............................................................................. 25

3  Tutorial: Building a Project. ...................................................................................... 27
3.1  Reviewing Concepts. ............................................................................................. 27
3.1.1  The Portfolio Valuation Sample Project. .......................................................... 28
3.1.2  Schema Discovery Using Input Adapters. ......................................................... 29
3.1.3  Simple Queries. ............................................................................................... 30
3.2  Creating the Sample Project. .............................................................................. 31
3.3  Editing a Project Diagram. ................................................................................... 32
3.4  Adding an Input Adapter. ...................................................................................... 33
3.5  Discovering a Schema. ....................................................................................... 35
3.6  Adding an Input Window Manually. .................................................................... 37
3.7  Creating an Aggregate as a Simple Query. .......................................................... 38
3.8  Creating a Join as a Simple Query. ...................................................................... 40
3.9  Preparing Data for Output. ................................................................................. 44
3.10  Adding an Output Adapter for SAP HANA. ........................................................ 45
3.11  Completing the Sample Project. ....................................................................... 46

4  Tutorial: Testing a Project. ......................................................................................... 48
4.1  Compiling the Sample Project. ............................................................................. 48
4.2 Viewing Problems. .............................................................. 49
4.3 Deploying the Sample Project. ........................................ 50
4.4 Loading Data into the Sample Project. .......................... 51
4.5 Testing the Project with Recorded Data. .......................... 52
4.6 Other Tools for Running and Testing Projects. ................. 53

5 Continuous Computation Language. ..................................... 54
  5.1 CCLScript. ................................................................. 54
  5.2 CCL Authoring. .......................................................... 55
  5.3 Editing in the CCL Editor. .............................................. 56
  5.4 CCL for the Sample Project. .......................................... 57
  5.5 CCL for Sample Project with Modules. .......................... 60
1 SAP Event Stream Processor Overview

SAP® Event Stream Processor enables you to create and run complex event processing applications to derive continuous intelligence from streaming event data in real time.

Event stream processing is a technique for analyzing information about events, in real time, for situational awareness. When vast numbers of event messages are flooding in, it is difficult to see the big picture. With event stream processing, you can analyze events as they stream in and identify emerging threats and opportunities as they happen. Event stream processing filters, aggregates, and summarizes data to enable better decision making based on more complete and timely information.

Event Stream Processor includes:

- A development platform, called ESP Studio, for building and testing event-based applications without significant programming effort.
- A run-time environment optimized for enterprise-scale, event-driven applications. Event Stream Processor processes data continually as it arrives, before storing it on disk, thus achieving extremely high throughput and low latency, enabling better decision making based on more complete and timely information.

Event Stream Processor does not replace databases. Unlike traditional databases, which are designed for on-demand queries and transaction processing, ESP is optimized for continuous queries. Thus, it complements traditional databases to help solve new classes of problems where continuous, event-driven data analysis is required.

So, even though Event Stream Processor can be installed and run as a standalone application, it is most commonly used as a component in a business solution based on SAP HANA. Not only can ESP look up information in SAP HANA tables and write data to SAP HANA tables, but the ESP Studio can be run from inside the SAP HANA studio.

SAP Event Stream Processor Deployments

Data flows into the SAP Event Stream Processor server from external sources through built-in or custom adapters, which translate incoming messages into a format that is accepted by the SAP Event Stream Processor server.

This figure shows a typical ESP deployment. Continuous queries, developed and tested as projects using the SAP Event Stream Processor Studio, are deployed to SAP Event Stream Processor server. Output adapters translate rows processed by SAP Event Stream Processor server into message formats that are compatible with external destinations such as SAP HANA, and send those messages downstream. SAP ESP Cockpit provides an operations console for configuring ESP.
Next Steps

Use this guide to:
- Learn key concepts
- Try out the development platform by building a simple project
- Watch a running application

1.1 Key Terms and Concepts

Events, projects, streams and windows, and continuous queries are the basics of data-flow programming in Event Stream Processor.

1.1.1 Events

A business event is a message that contains information about an actual business event that occurred. Many business systems produce streams of such events as things happen.

With SAP Event Stream Processor, through SAP Event Stream Processor Studio or CCL, you can use streams, windows, and keyed streams with adapters to create complex projects. Streams, windows, and keyed streams allow you to consume and process input events and generate output events.
Using CCL, you can also include delta streams in your projects. SAP recommends delta streams for advanced users only.

Examples of business events that are often transmitted as streams of event messages include:

- Financial market data feeds that transmit trade and quote events, where each event may consist of ticket symbol, price, quantity, time, and so on.
- Radio Frequency Identification System (RFID) sensors that transmit events indicating that an RFID tag was sensed nearby.
- Electronic sensors that transmit messages indicating the health of remote equipment and its components.
- Click streams, which transmit a message (a click event) each time a user clicks a link, button, or control on a Web site.
- Database transaction events, which occur each time a record is added to a database or updated in a database.

**Event Blocks**

Business events can be published into ESP projects in collections called Event Blocks, improving the performance of your ESP projects. Event blocks come in two different types: envelopes and transactions. As an event block is being processed by a window, resulting rows are not sent downstream immediately. Instead, they are stored until the last event of the block is processed, and the resulting events are then sent downstream. Event blocks have the following properties:

- **Envelopes:**
  - Each row in an envelope is treated atomically; a failure in an event does not discard the envelope. This behavior is useful if a model’s performance is important, but not necessarily the integrity of the data.

- **Transactions:**
  - A transaction will be discarded if any one event in the block fails. This behavior can be used to guarantee that logical blocks of events are completely error-free.
  - Before a transaction block is sent downstream, all events in the transaction are compressed as much as possible. For example, an event with an insert and then an update will compress down to a single insert with updated values.

**1.1.2 Operation Codes**

The operation code (opcode) of an event record specifies the action to perform on the underlying store of a window for that event.

In many Event Stream Processor use cases, events are independent of each other: each carries information about something that happened. In these cases, a stream of events is a series of independent events. If you define a window on this type of event stream, each incoming event is inserted into the window. If you think of a window as a table, the new event is added to the window as a new row.
In other use cases, events deliver new information about previous events. ESP needs to maintain a current view of the set of information as the incoming events continuously update it. Two common examples are order books for securities in capital markets, and open orders in a fulfillment system. In both applications, incoming events may indicate the need to:

- Add an order to the set of open orders,
- Update the status of an existing open order, or,
- Remove a canceled or filled order from the set of open orders.

To handle information sets that are updated by incoming events, Event Stream Processor recognizes the following opcodes in incoming event records:

- **insert**  
  Insert the event record.

- **update**  
  Update the record with the specified key. If no such record exists, it is a runtime error.

- **delete**  
  Delete the record with the specified key. If no such record exists, it is a runtime error.

- **upsert**  
  If a record with a matching key exists, update it. If a record with a matching key does not exist, insert this record.

- **safedelete**  
  If a record with a matching key exists, delete it. If a record with a matching key does not exist, do nothing.

All event records include an opcode. Each stream or window in the project accepts incoming event records and outputs event records. Output events, including opcodes, are determined by their source (window, stream, keyed stream, or delta stream) and the processing specified for it.

Refer to the *SAP Event Stream Processor: Developer Guide* for details on how windows and streams interpret the opcodes on incoming event records and generate opcodes for output records.

### 1.1.3 Streams and Windows

Both streams and windows process events. The difference is that windows have state, meaning they can retain and store data, while streams are stateless and cannot.

Streams process incoming events and produce output events according to the continuous query that is attached to the stream, but no data is retained.

A window consists of a table where incoming events can add rows, update existing rows, or delete rows. You can set the size of the window based on time, or on the number of events recorded. For example, a window might retain all events over the past 20 minutes, or the most recent 1,000 events. A window can also retain all events. In this case, the incoming event stream must be self-managing in that it contains events that both insert rows into the window and delete rows from the window, so that the window does not grow infinitely large. Windows are needed for performing aggregate operations, as this cannot be done on streams.
1.1.4 Streams, Windows, Adapters, and Continuous Queries in Projects

ESP projects are like applications, and consist of a set of event streams, any other required datasources, and the business logic applied to incoming event data to produce results.

At its most basic level, a project consists of:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input streams and windows</td>
<td>Receives input data flowing into the project. An input stream can receive incoming event data on an event-driven basis, and can also receive static or semistatic sets of data that are loaded once or periodically refreshed. Input streams that have state—that is, they can retain and store data—are called windows.</td>
</tr>
<tr>
<td>Adapters</td>
<td>Connects an input stream or window to a datasource. ESP includes a large set of built-in adapters as well as an SDK that you can use to build custom adapters. Adapters can also connect an output stream or window to a destination. While an adapter connects the project to external inputs and outputs, technically it is not part of the project.</td>
</tr>
<tr>
<td>Derived streams and windows</td>
<td>Takes data from one or more streams or windows and applies a continuous query to produce a new stream or window. Derived streams that have state are windows.</td>
</tr>
</tbody>
</table>

1.1.5 Data-Flow Programming

SAP® Event Stream Processor uses data-flow programming for processing event streams.

In data-flow programming, you define a set of event streams and the connections between them, and apply operations to the data as it flows from sources to outputs.

Data-flow programming breaks a potentially complex computation into a sequence of operations with data flowing from one operation to the next. This technique also provides scalability and potential parallelization, since each operation is event driven and independently applied. Each operation processes an event only when it is received from another operation. No other coordination is needed between operations.

The sample project shown in the figure shows a simple example of this.

Each of the continuous queries in this simple example—the VWAP aggregate, the IndividualPositions join object, and the ValueByBook aggregate—is a type of derived stream, as its schema is derived from other inputs in the diagram, rather than originating directly from external sources. You can create derived streams in a diagram using the simple query elements provided in the Studio Visual editor, or by defining your own explicitly.
### Table 1: Data-Flow Diagram Contents

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PriceFeed</td>
<td>Represents an input window, where incoming data from an external source complies with a schema consisting of five columns, similar to a database table with columns. The difference is that in Event Stream Processor, the streaming data is not stored in a database.</td>
</tr>
<tr>
<td>Positions</td>
<td>Another input window, with data from a different external source. Both Positions and PriceFeed are included as windows, rather than streams, so that the data can be aggregated.</td>
</tr>
<tr>
<td>VWAP</td>
<td>Represents a simple continuous query that performs an aggregation, similar to a SQL Select statement with a GROUP BY clause.</td>
</tr>
<tr>
<td>Element</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IndividualPositions</td>
<td>Represents a simple continuous query that performs a join of Positions and VWAP, similar to a SQL FROM clause that produces a join.</td>
</tr>
<tr>
<td>ValueByBook</td>
<td>Another simple query that aggregates data from the stream Individual Positions.</td>
</tr>
</tbody>
</table>

**Related Information**

The Portfolio Valuation Sample Project [page 28]

The Portfolio Valuation project that you build in this tutorial applies current prices to a portfolio of investments to compute the value of each investment and of the portfolio. It uses simple queries to aggregate and join data from two input windows, then outputs the refined data into SAP HANA using an output adapter.

Diagrams [page 21]

In visual authoring, you use diagrams to create and manipulate the streams, windows, connections, and other components of a project, and create simple queries.

**1.1.6 Getting Results from a Project**

SAP Event Stream Processor has four ways to get output from a running project.

- Applications receive information automatically from output adapters attached to a stream when you build the project. For example, you can load data into an SAP HANA database using an output adapter.
- Applications can subscribe to data streams by means of an external subscriber, which users can create using subscription APIs provided with the product.
- You can start a new project that binds (connects) to a stream in a running project, without reconfiguring the project.
- You can run on-demand queries against output windows in a running ESP project. This is similar to querying a database table.
  - From the command line, using the `streamingquery` tool. For more information, see the SAP Event Stream Processor: Utilities Guide.
1.1.7 SAP HANA

SAP HANA is an in-memory database and platform for business applications.

SAP HANA is an in-memory, columnar, massively parallel processing platform that provides a common database for online transactional processing (OLTP) and online analytical processing (OLAP), eliminating redundancy and latency. It also unifies end-to-end data processing with search, text, geospatial and predictive analytics on top of the in-memory columnar foundation, further extending the benefit of redundancy and latency elimination to all data processing workloads.

SAP HANA can be deployed as a managed service in the cloud or as an optimized on-premise appliance with tailored data center integration.

1.2 User Roles and Capabilities

Application developers use the SAP Event Stream Processor Studio, including the visual editor, text editor, and testing tools, to develop event-based applications. You can also develop custom adapters using the the adapter toolkit or the SDKs, and monitor and manage your projects using the SAP ESP Cockpit.

ESP Studio is available as a plugin for SAP HANA studio, or as a standalone product. Most HANA-specific features noted in the table are only available when working in the SAP HANA studio.
<table>
<thead>
<tr>
<th>User</th>
<th>What You Can Do</th>
</tr>
</thead>
</table>
| Application developer  | Use Studio to create sophisticated data-flow applications in the visual editor, a graphical authoring environment.  
Use the text editor in Studio to develop projects in Continuous Computation Language (CCL). Switch between the fully integrated editors, and see changes in one editor immediately reflected in the other.  
Access SAP HANA databases using adapters, smart data access, reference table queries, or the SAP HANA navigator.  
Augment data from SAP HANA platform, and various other sources, using custom applications.  
Create custom operators and external functions by embedding CCLScript scripts in your CCL code or diagram.  
Test compiled projects by running them on a local or remote server in the SAP ESP Run-Test perspective.  
Watch data flow through the project, record and play back in-flowing data, trace events, set breakpoints and watch variables on stream inputs and outputs, monitor performance, execute continuous and on-demand queries, and more.  
Create, run, and test projects using command-line tools as an alternative to Studio. |
| Adapter developer      | Create custom input and output adapters using the ESP adapter toolkit, or one of the SDKs provided with Event Stream Processor.  
Integrate custom function libraries using the SDKs.                                                                                                                                                     |
| Business analyst       | Respond to business events in real time, using ESP analytics and alerts in conjunction with SAP HANA data.  
Design continuous queries in the Studio visual editor with minimal knowledge of programming.  
Run projects in Studio before production deployment, to ensure that they satisfy your business requirements.  
Issue on-demand queries on a running ESP project.                                                                                                                                                    |
<table>
<thead>
<tr>
<th>User</th>
<th>What You Can Do</th>
</tr>
</thead>
</table>

### 1.3 Beyond the Basics

ESP projects can take advantage of a broad set of features.

In addition to the basic elements of input streams, adapters, and output streams, a project may include:

- Derived streams
- References to database tables
- CCLScript code
- External C code
- Custom adapters
- Modules that can be developed independently and loaded into a project
- Named schemas that store reusable schema definitions

See the SAP Event Stream Processor: Developer Guide and the SAP Event Stream Processor: CCL Reference for code samples that demonstrate CCL and CCLScript.


### Related Information

**CCLScript** [page 54]

CCLScript is a scripting language that brings extensibility to CCL, allowing you to create custom operators and functions that go beyond standard SQL.

**CCL for Sample Project with Modules** [page 60]

*This variation of the portfolio valuation project uses a defined module with a named schema to easily scale out the application in a very high volume deployment. The SAP HANA Output adapter is not included in this module.*
2  Getting Started with a Project

To begin developing a project, start the SAP Event Stream Processor Studio, review Studio basics, and optionally step through an example before creating your own project.

2.1  Launching Studio

Start SAP Event Stream Processor Studio using the desktop shortcut, Windows Start menu, or the command line. If you installed the ESP plugin for SAP HANA studio, you can access SAP Event Stream Processor functionality within the SAP HANA studio by opening the SAP ESP Run-Test and SAP ESP Authoring perspectives.

Procedure

1. In SAP HANA studio, go to Window ➔ Open Perspective ➔ Other, then open the SAP ESP Authoring or SAP ESP Run-Test perspective.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>In SAP HANA studio, go to Window ➔ Open Perspective ➔ Other, then open the SAP ESP Authoring perspective or the SAP ESP Run-Test perspective. From outside of SAP HANA studio, either: ○ Double-click the SAP ESP Studio shortcut on your computer desktop, or, ○ Select Event Stream Processor Studio from the Start menu.</td>
</tr>
<tr>
<td>Linux or UNIX</td>
<td>In SAP HANA studio, go to Window ➔ Open Perspective ➔ Other, then open the SAP ESP Authoring perspective or the SAP ESP Run-Test perspective. From outside of SAP HANA studio, either: ○ Double-click the SAP ESP Studio shortcut on your computer desktop, or, ○ At the command line, enter STREAMING_HOME/studio/streamingstudio.</td>
</tr>
</tbody>
</table>

Related Information

Exploring Studio Perspectives and Views [page 15]
Explore SAP Event Stream Processor Studio perspectives and views to discover what you can do.

The Authoring Perspective [page 17]
The visual editor, CCL editor, and other tools and views in the SAP ESP Authoring perspective allow you to create, view, and edit a diagram or CCL file.

The Run-Test Perspective [page 19]
In the SAP ESP Run-Test perspective, you access tools to test, monitor, debug, and fine-tune a project.
Diagrams [page 21]
In visual authoring, you use diagrams to create and manipulate the streams, windows, connections, and other components of a project, and create simple queries.

Learning through Sample Projects [page 22]
SAP Event Stream Processor Studio includes several example projects.

Project Execution and Testing [page 25]
Run and test all aspects of a project using SAP Event Stream Processor Studio.

2.2 Exploring Studio Perspectives and Views

Explore SAP Event Stream Processor Studio perspectives and views to discover what you can do.

Context
Use the sample projects to see examples of different project structures and diagrams in the Visual editor.

Procedure

1. (Optional) If you are accessing SAP Event Stream Processor functionality using the Event Stream Processor plugin for SAP HANA studio, set up the SAP HANA studio Welcome screen to view various resources.
   a. Click the Customize Page icon to open the Customize dialog.
   b. Select the Home tab.
   c. In the Root Pages section, select the resources you would like to appear on the Welcome screen. Click OK to save.
      For example,
      ○ Click Overview to access SAP Event Stream Processor documentation and an overview of the features provided by Maven Integration for Eclipse.
      ○ Click Tutorials to access cheat sheet guides on how to create and run a project for SAP ESP.
      ○ Click Samples to access examples of completed projects that you can explore and run.

2. (Optional) On the Welcome screen, use the tabs to navigate to the help, or close the Welcome screen tab.
   ○ Click Overview to access SAP Event Stream Processor documentation and an overview of the features provided by Maven Integration for Eclipse.
   ○ Click Tutorials to access cheat sheet guides on how to create and run an SAP ESP Project.
   ○ Click Samples to access examples of completed projects that you can explore and run.
   ○ Click What’s New to access the latest updates.
   ○ Click Web Resources to access online videos and to join the SAP ESP Developer Centre, which allows you to stay informed about SAP Event Stream Processor.
   ○ Click Workbench to open Studio.
3. To open ESP perspectives, go to Window > Open Perspective > Other, then open the SAP ESP Authoring perspective or the SAP ESP Run-Test perspective. To switch to another perspective, click its tab just below the main menu bar.

   a. Use the SAP ESP Authoring perspective to:
      - Create and edit projects
      - Develop projects and diagrams in the Visual editor, a graphical editing environment
      - Develop projects in the CCL editor, a text-oriented editing environment where you edit CCL code
      - Compile projects

   b. Use the SAP ESP Run-Test perspective to:
      - Connect to servers
      - Run projects
      - Enter test data by uploading data files to a server, or entering data manually to a stream
      - Publish data
      - Execute a query against a running project
      - Use the Event Tracer and Debugger to set breakpoints and watchpoints, and trace the flow of data through a project
      - Record incoming event data to a playback file, and play back captured data into a running project
      - Monitor performance

Results

For more information, see the SAP Event Stream Processor: Studio Users Guide.

Related Information

Launching Studio [page 14]
Start SAP Event Stream Processor Studio using the desktop shortcut, Windows Start menu, or the command line. If you installed the ESP plugin for SAP HANA studio, you can access SAP Event Stream Processor functionality within the SAP HANA studio by opening the SAP ESP Run-Test and SAP ESP Authoring perspectives.

The Authoring Perspective [page 17]
The visual editor, CCL editor, and other tools and views in the SAP ESP Authoring perspective allow you to create, view, and edit a diagram or CCL file.

The Run-Test Perspective [page 19]
In the SAP ESP Run-Test perspective, you access tools to test, monitor, debug, and fine-tune a project.

Diagrams [page 21]
In visual authoring, you use diagrams to create and manipulate the streams, windows, connections, and other components of a project, and create simple queries.

Learning through Sample Projects [page 22]
SAP Event Stream Processor Studio includes several example projects.

Project Execution and Testing [page 25]
Run and test all aspects of a project using SAP Event Stream Processor Studio.

### 2.3 The Authoring Perspective

The visual editor, CCL editor, and other tools and views in the SAP ESP Authoring perspective allow you to create, view, and edit a diagram or CCL file.

When viewing the SAP ESP Authoring perspective, its components are:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editor</td>
<td>(Center) Authoring perspective where you edit the diagram (in the visual editor) or CCL (in the CCL editor). The Visual and CCL text editors are completely integrated. When you save and switch to the other editor, your work is saved there as well.</td>
</tr>
<tr>
<td>Palette</td>
<td>(Right side of Editor) Includes groups of tools used to create new CCL elements on the diagram. Most shapes on the Palette correspond to a CCL statement.</td>
</tr>
<tr>
<td>Project Explorer</td>
<td>(Left) Provides a hierarchical tree structure of folders and files.</td>
</tr>
<tr>
<td>Properties view</td>
<td>(Bottom right) Displays the properties of the object selected in the diagram. You can also set properties in this view, and edit expressions.</td>
</tr>
<tr>
<td>Outline view</td>
<td>(Top right) Provides an index to all elements in the diagram as a hierarchical tree structure. Also shows the order in which adapters are started. Right-click an element in this view to show it in the diagram, delete it, modify it, or add a child element.</td>
</tr>
<tr>
<td>Overview</td>
<td>(Right) Helps you understand the big picture, and navigate easily to different areas of a large, complex diagram. For large diagrams you can scroll the editor by dragging the gray box in the overview.</td>
</tr>
<tr>
<td>Search</td>
<td>(Top center) Provides full-text search capability for finding text strings in the workspace. Useful in navigating File Explorer, and project contents in the CCL editor. You can filter search results, and copy, remove, or replace results found.</td>
</tr>
<tr>
<td>Problems</td>
<td>(Below Editor, leftmost tab) Displays errors found when you compile a project.</td>
</tr>
<tr>
<td>Console</td>
<td>(Below Editor, second tab) Displays messages generated when interacting with ESP components.</td>
</tr>
</tbody>
</table>
You can customize the arrangement of views in your perspectives. See Customizing the Studio Work Environment in the SAP Event Stream Processor: Studio Users Guide.

Related Information

Launching Studio [page 14]
Start SAP Event Stream Processor Studio using the desktop shortcut, Windows Start menu, or the command line. If you installed the ESP plugin for SAP HANA studio, you can access SAP Event Stream Processor functionality within the SAP HANA studio by opening the SAP ESP Run-Test and SAP ESP Authoring perspectives.

Exploring Studio Perspectives and Views [page 15]
Explore SAP Event Stream Processor Studio perspectives and views to discover what you can do.

The Run-Test Perspective [page 19]
In the SAP ESP Run-Test perspective, you access tools to test, monitor, debug, and fine-tune a project.

Diagrams [page 21]
In visual authoring, you use diagrams to create and manipulate the streams, windows, connections, and other components of a project, and create simple queries.

Learning through Sample Projects [page 22]
SAP Event Stream Processor Studio includes several example projects.

Project Execution and Testing [page 25]
Run and test all aspects of a project using SAP Event Stream Processor Studio.
### 2.4 The Run-Test Perspective

In the SAP ESP Run-Test perspective, you access tools to test, monitor, debug, and fine-tune a project.

When viewing the SAP ESP Run-Test perspective, its components are:

1. **Server View** (Upper left) Start and connect to available servers. Your first project is there, already running.
2. **Manual Input view** (Below Server View, leftmost tab) Manually create and publish events as input to a stream or window.
3. **Playback view** (Below Server View, second tab) Record data flowing into a running project, or play back recorded files.
4. **File Upload view** (Below Server View, third tab) Publish an existing data file to an input stream or window.
5. **SQL Query view** (Below Server View, right tab) Run a snapshot SQL query. It captures a snapshot of the current window state and displays results in the Console.
6. **Console view** (Lower right) Review log messages and other tracing and debugging information useful to developers.
7. **Stream view** (Middle, leftmost tab) Show the events of an output stream or the retained events in an output window of a running project.
8. **Monitor view** (Tabbed to the right of Stream view) Monitor performance of a running project.
9. **Debugger view** (Tabbed to the right of Stream view) Debug a project by setting breakpoints and watchpoints.
10. **Event Tracer view** (Tabbed to the right of Stream view by default) Trace the flow of data through a project.

Other Run-Test tools include:
- Run Project button, which is also in the SAP ESP Authoring perspective.

The figure shows the sample project running in SAP ESP Run-Test perspective.
Launching Studio [page 14]
Start SAP Event Stream Processor Studio using the desktop shortcut, Windows Start menu, or the command line. If you installed the ESP plugin for SAP HANA studio, you can access SAP Event Stream Processor functionality within the SAP HANA studio by opening the SAP ESP Run-Test and SAP ESP Authoring perspectives.

Exploring Studio Perspectives and Views [page 15]
Explore SAP Event Stream Processor Studio perspectives and views to discover what you can do.

The Authoring Perspective [page 17]
The visual editor, CCL editor, and other tools and views in the SAP ESP Authoring perspective allow you to create, view, and edit a diagram or CCL file.

Diagrams [page 21]
In visual authoring, you use diagrams to create and manipulate the streams, windows, connections, and other components of a project, and create simple queries.

Learning through Sample Projects [page 22]
SAP Event Stream Processor Studio includes several example projects.

Project Execution and Testing [page 25]
Run and test all aspects of a project using SAP Event Stream Processor Studio.
2.5 Diagrams

In visual authoring, you use diagrams to create and manipulate the streams, windows, connections, and other components of a project, and create simple queries.

When you open a project in the Visual editor, the project shows a collection of stream and window shapes that are connected with arrows showing the flow of data. You develop the project by selecting new input and output streams, windows, and other elements from the Palette, dropping them onto the diagram, connecting them, and configuring their behavior.

Every project has at least one diagram. A diagram in the Visual editor is a projection of the associated CCL statements in the project.

When you add a shape or other element to a diagram, it is automatically added to the project when you save. You can delete an element from a diagram only, or from the project.

Display diagrams in verbose or iconic mode:

- **iconic**: compartments are collapsed to save space.

  ![Iconic Diagram](image)

- **verbose**: all compartments in elements are visible.

  ![Verbose Diagram](image)

  - To expand or collapse all shapes in the diagram, use the All Verbose or All Iconic buttons on the main toolbar.
  - To expand an individual shape, select it and click the "+" box in the shape toolbar.
  - To collapse an individual shape, select it and click the "-" box in the shape toolbar.
Related Information

Launching Studio [page 14]
Start SAP Event Stream Processor Studio using the desktop shortcut, Windows Start menu, or the command line. If you installed the ESP plugin for SAP HANA studio, you can access SAP Event Stream Processor functionality within the SAP HANA studio by opening the SAP ESP Run-Test and SAP ESP Authoring perspectives.

Exploring Studio Perspectives and Views [page 15]
Explore SAP Event Stream Processor Studio perspectives and views to discover what you can do.

The Authoring Perspective [page 17]
The visual editor, CCL editor, and other tools and views in the SAP ESP Authoring perspective allow you to create, view, and edit a diagram or CCL file.

The Run-Test Perspective [page 19]
In the SAP ESP Run-Test perspective, you access tools to test, monitor, debug, and fine-tune a project.

Learning through Sample Projects [page 22]
SAP Event Stream Processor Studio includes several example projects.

Project Execution and Testing [page 25]
Run and test all aspects of a project using SAP Event Stream Processor Studio.

Data-Flow Programming [page 8]
SAP® Event Stream Processor uses data-flow programming for processing event streams.

2.6 Learning through Sample Projects

SAP Event Stream Processor Studio includes several example projects.

The Welcome page in ESP Studio provides samples of completed projects that you can explore and run. If you are using the Event Stream Processor plugin for the SAP HANA studio, the examples do not display by default on the Welcome page. Instead, customize the Welcome page in SAP HANA studio to show the samples. Click the Customize Page icon in the top right corner of the Welcome page to open the Customize dialog. Select the Home tab and in the Root Pages section, select the Samples option and click OK. The Welcome page now displays a Samples subpage.

You can view the examples in Studio and run them against sample data installed with the product. Stepping through examples is an ideal way to watch a simplified set of event data flow through the system.

The examples include:

- **Indexes Calculation**
  Shows how continuous computations can be applied to a stream of market prices to deliver insight into the market. This example demonstrates reusable modules. Each of the market calculations is defined in an external module, that is, a module defined in a separate CCL file, and then imported into the project. Parameters (in this case, time and intervals) are set when the module is called.

- **Pattern Matching**
  Provides a simple example of situation detection: watching for a pattern of events. The example demonstrates how to watch for employee fraud in a retail setting, based on transaction patterns
from a point-of-sale system. The example applies three Filter queries to an input stream of transactions, and then uses a Pattern query (CCL MATCHING clause) to produce a Possible Fraud Alert event when all of the criteria occur in the defined time interval.

**Prepay Biller**
Loads call events (CDRs), such as those generated from a telephone carrier network, and applies account and call plan information to create a billing record for each call, and maintain a balance of prepaid minutes. The example demonstrates joins, aggregation, and using joins to augment the events with reference data.

**Top 3 Prices**
Creates a window showing the top three distinct trade prices for each symbol. The example uses a Flex operator to create a custom operator with an embedded CCLScript script. A Flex operator creates a single output stream or window, and allows greater flexibility and control than a simple SELECT statement. The example also uses a named schema, which can be defined once and shared by the input stream and output window.

**VWAP**
Defines an input stream of stock market prices, as they might be reported from an exchange, and computes a moving average price called the volume weighted average price (VWAP). Uses a filter, and a simple aggregation (GROUP BY).

For details of each example, click **Samples** on the SAP Event Stream Processor Welcome page.

See the SAP Event Stream Processor: Developer Guide and the SAP Event Stream Processor: CCL Reference for code samples that demonstrate CCL and CCLScript.

---

**Related Information**

**Launching Studio** [page 14]
Start SAP Event Stream Processor Studio using the desktop shortcut, Windows Start menu, or the command line. If you installed the ESP plugin for SAP HANA studio, you can access SAP Event Stream Processor functionality within the SAP HANA studio by opening the SAP ESP Run-Test and SAP ESP Authoring perspectives.

**Exploring Studio Perspectives and Views** [page 15]
Explore SAP Event Stream Processor Studio perspectives and views to discover what you can do.

**The Authoring Perspective** [page 17]
The visual editor, CCL editor, and other tools and views in the SAP ESP Authoring perspective allow you to create, view, and edit a diagram or CCL file.

**The Run-Test Perspective** [page 19]
In the SAP ESP Run-Test perspective, you access tools to test, monitor, debug, and fine-tune a project.

**Diagrams** [page 21]
In visual authoring, you use diagrams to create and manipulate the streams, windows, connections, and other components of a project, and create simple queries.

**Project Execution and Testing** [page 25]
Run and test all aspects of a project using SAP Event Stream Processor Studio.

**The Portfolio Valuation Sample Project** [page 28]
The Portfolio Valuation project that you build in this tutorial applies current prices to a portfolio of investments to compute the value of each investment and of the portfolio. It uses simple queries to aggregate and join data from two input windows, then outputs the refined data into SAP HANA using an output adapter.
Tutorial: Building a Project [page 27]

Walk through this hands-on tutorial to create a simple project in the Visual editor.

2.6.1 How to load a Sample Project

Load and run one of the example projects installed with the product, so that you can view end-to-end project execution in your workspace.

Prerequisites

To run these examples, you may need to disable McAfee host intrusion prevention. See your McAfee documentation for details.

If you are using the Event Stream Processor plugin for the SAP HANA studio, first click the Customize Page icon in the top right corner of the Welcome page to open the Customize dialog. Select the Home tab and in the Root Pages section, select the Samples option and click OK. Now you can access the example projects from the Samples subpage on the Welcome page.

Procedure

1. Navigate to the Studio Welcome page.
2. Select Samples.
3. Look at the samples and read their descriptions.
4. Click the name of the sample you wish to load into your workspace. The SAP ESP Authoring perspective opens and loads the example.
   Project Explorer shows all example projects, plus any other projects you have created.
5. Click in the main toolbar to run the project.

   Note

   If you are prompted for a user name and password for the local cluster, you must first change the default system-generated credentials. See SAP Event Stream Processor: Studio Users Guide > Troubleshooting > Prompted for Local Cluster Password.

6. The sample opens and publishes data automatically, in the SAP ESP Run-Test perspective. Look at the various elements in the perspective to learn about the process.
   ○ Server View shows localhost and connection information, as well as the sample project, which is now subscribed to the example stream.
   ○ The Console shows a series of status messages detailing server activity.
7. Open windows and streams to view published data in Stream View.
   a. In Server View, double-click each output window or output stream.
Each stream or window opens in a separate tab. For example, the IndexesCalculation example opens four tabs, with one input stream and three output windows.

b. Double-click and open input windows and input streams to view input data.

Next Steps

Run a second sample project. Server View now shows both samples. Expand it to show all streams for each project.

2.7 Project Execution and Testing

Run and test all aspects of a project using SAP Event Stream Processor Studio.

During development, you can use Studio to run any compiled project against a local or remote server, view data flowing through the streams and windows defined in the project, execute queries, and use debugging tools. Your project configuration and licensing determine the type of server connections you can use when running projects. Some adapters also have special licensing requirements.

In Studio you can connect immediately to a local cluster to run projects, using default security established for Studio during installation. A cluster consists of a group of server nodes, which are processes that run on hosts. A cluster can have a single node or multiple nodes.

In a production environment, you typically run projects on a remote server. Administrators monitor and manage ESP server nodes, clusters, and projects using SAP ESP Cockpit. Command-line utilities and procedures are also available, as discussed in the SAP Event Stream Processor: Configuration and Administration Guide.

Related Information

Launching Studio [page 14]
Start SAP Event Stream Processor Studio using the desktop shortcut, Windows Start menu, or the command line. If you installed the ESP plugin for SAP HANA studio, you can access SAP Event Stream Processor functionality within the SAP HANA studio by opening the SAP ESP Run-Test and SAP ESP Authoring perspectives.

Exploring Studio Perspectives and Views [page 15]
Explore SAP Event Stream Processor Studio perspectives and views to discover what you can do.

The Authoring Perspective [page 17]
The visual editor, CCL editor, and other tools and views in the SAP ESP Authoring perspective allow you to create, view, and edit a diagram or CCL file.

The Run-Test Perspective [page 19]
In the SAP ESP Run-Test perspective, you access tools to test, monitor, debug, and fine-tune a project.

Diagrams [page 21]
In visual authoring, you use diagrams to create and manipulate the streams, windows, connections, and other components of a project, and create simple queries.

**Learning through Sample Projects** [page 22]

*SAP Event Stream Processor Studio includes several example projects.*

**Tutorial: Testing a Project** [page 48]

*Compile, run, and test the project you have created by using tools in the SAP ESP Run-Test perspective.*
3 Tutorial: Building a Project

Walk through this hands-on tutorial to create a simple project in the Visual editor.

Context

The sample project demonstrates how you can easily define event streams and windows by attaching a previously configured adapter and discovering its schema, or by manually defining a window and its schema. It shows you how to define continuous queries—aggregations, joins, and more—using the visual tools.

Begin by reviewing background information that helps you understand the project. Then complete the tasks to build the project.

1. Reviewing Concepts [page 27]
   Begin the tutorial by reading a description of the sample project, and concepts applied in it.

2. Creating the Sample Project [page 31]
   Define a new set of processing instructions for event data.

3. Editing a Project Diagram [page 32]
   Edit projects in the Visual editor by adding shapes from the Palette to the project diagram, connecting them, and completing the configuration of each shape.

4. Adding an Input Adapter [page 33]
   Attach an adapter by inserting it in the diagram, connecting it to a stream or window, and setting properties.

5. Discovering a Schema [page 35]
   Use the Schema Discovery button in the Adapter shape to discover and automatically create a schema based on the format of the data from the adapter.

6. Adding an Input Window Manually [page 37]
   Add an input window to the diagram in the sample my_portfolio_valuation project.

7. Creating an Aggregate as a Simple Query [page 38]
   Add an Aggregate simple query to the sample diagram to create a volume weighted average price (VWAP).

8. Creating a Join as a Simple Query [page 40]
   Add a join to the sample project. A join combines events from two or more inputs to create a single stream or window. It is similar to a join in SQL.

9. Preparing Data for Output [page 44]
   Add a new aggregate to finalize the data and prepare it for output into SAP HANA.

10. Adding an Output Adapter for SAP HANA [page 45]
    Create a service definition for an ODBC connection to an SAP HANA database, and insert and configure a HANA Output adapter.

11. Completing the Sample Project [page 46]
    Clean up the diagram by removing unused elements.

3.1 Reviewing Concepts

Begin the tutorial by reading a description of the sample project, and concepts applied in it.
3.1.1 The Portfolio Valuation Sample Project

The PortfolioValuation project that you build in this tutorial applies current prices to a portfolio of investments to compute the value of each investment and of the portfolio. It uses simple queries to aggregate and join data from two input windows, then outputs the refined data into SAP HANA using an output adapter.

The example:

1. Receives a stream of prices in an input window called PriceFeed. The schema for this window has five columns: Id, Symbol, Price, Shares, and TradeTime. The window uses the Id field as a primary key, and is set to keep the last 10 minutes of price events.

2. Applies an Aggregate simple query to create a 10-minute moving average—a volume weighted average price (VWAP). With the VWAP, you can see the value of positions based on the average price, rather than see the value of your positions change with every small price movement. The VWAP formula is calculated as:

   \[
   \frac{\text{sum(PriceFeed.Price} \times \text{PriceFeed.Shares)}}{\text{sum(PriceFeed.Shares)}}
   \]

3. Reads data from another input window, Positions, with three columns: BookId, Symbol, and SharesHeld.

4. Applies a Join simple query, joining the market price (from the VWAP aggregate) to your holdings (Positions), so that you can see the value of your position in each stock:

   FROM VWAP
   RIGHT JOIN Positions
   ON VWAP.Symbol = Positions.Symbol

5. Applies one more aggregation to show the total value of each "book." This aggregate, ValueByBook, groups current and average values for individual positions into different "books." Each book may comprise a set of investment portfolios or funds. In the CCL, a GROUP BY clause performs the aggregation:

   CREATE OUTPUT WINDOW ValueByBook
   PRIMARY KEY DEDUCED
   AS
   SELECT IndividualPositions.BookId BookId, 
       sum(IndividualPositions.AveragePosition) AveragePosition
   FROM IndividualPositions
   GROUP BY IndividualPositions.BookId ;

6. Attaches an output adapter to output the processed data to an SAP HANA table. This adapter requires an existing connection to SAP HANA, created using the Data Services view.

   ATTACH OUTPUT ADAPTER toSAPHANA TYPE hana_out
   TO ValueByBook
   PROPERTIES service = 'hanaservice',
   table = 'exampletable',
   dataWarehouseMode = 'ON',
   msdateColumnName = 'TIME';
Related Information

CCL for the Sample Project [page 57]

The CCL for the Portfolio Valuation sample project created in the Studio Visual editor is shown here, with the corresponding shape in the diagram for each element. Line breaks are added for readability.

3.1.2 Schema Discovery Using Input Adapters

In the tutorial you use the schema discovery feature to discover an external schema and create a CCL schema based on the format of the data from the datasource connected to an adapter.

Input Adapters in the Diagram

An input adapter identifies the external source for the input stream or window, and translates it into a format that Event Stream Processor accepts. You can add adapters to the diagram before or after adding input and output streams or windows.

ESP provides a set of built-in adapters for common databases, message bus, file systems, sockets, and more. You can also develop custom adapters using an SDK. Most adapters provided can be used as input or output adapters.

Schema Discovery Basics

Every row in a stream or window must have the same structure, or schema, which includes the column names, the column datatypes, and the order in which the columns appear. Multiple streams or windows can use the same schema, but each stream or window can only have one schema.
Rather than manually creating a new schema in your ESP project, you can use schema discovery to discover and automatically create a schema, stream, or window based on the format of the data from the datasource to which your adapter connects. For example, you create a table in your SAP HANA database and use the SAP HANA Output adapter to connect to the database. You can then use schema discovery to discover and create a schema, stream, or window in your ESP project that corresponds to the schema of the table you created in your SAP HANA database.

In the tutorial, you discover the schema for the PriceFeed input window from the File/Hadoop Event XML Input adapter.

To discover a schema, you need to first configure the adapter properties. Each adapter that supports schema discovery has unique properties that must be set to enable schema discovery. For a list of adapters that support schema discovery and properties to configure, see the SAP Event Stream Processor: Studio Users Guide. For property details, see your adapter type in the SAP Event Stream Processor: Adapters Guide.

Related Information

Adding an Input Adapter [page 33]
Attach an adapter by inserting it in the diagram, connecting it to a stream or window, and setting properties.

Discovering a Schema [page 35]
Use the Schema Discovery button in the Adapter shape to discover and automatically create a schema based on the format of the data from the adapter.

3.1.3 Simple Queries

Accomplish most common querying tasks using a set of simple queries available in the Visual editor: filter, aggregate, join, compute, union, and pattern.

The tools for these six queries are available as objects in the Palette, in Streams and Windows.

*Filter* allows you to filter a stream down to only the events of interest, based on a filter expression. Similar to SQL WHERE clause.

*Aggregate* allows you to group events that have common values and compute summary statistics for the group, such as an average. You can also define a window size, based on either time or number of events. Uses the CCL GROUP BY clause, similar to SQL GROUP BY.

*Join* allows you to combine records from multiple streams or windows, forming a new record with information from each source. Comparable to a join in SQL, where you specify two or more sources in the FROM clause.

*Compute* allows you to create a new event with a different schema, and compute the value to be contained in each column (field) of the new event. Comparable to a projection in SQL, where you use a SELECT statement to specify the column expressions, and FROM to specify a single source.

*Union* allows you to combine multiple streams or windows that all share a common schema into a single stream or window. Similar to SQL UNION operator.
Lets you watch for patterns of events within a single stream or window or across multiple streams and windows. When ESP detects an event pattern in a running project, it produces an output event. This uses the CCL MATCHING clause.

Related Information

Creating an Aggregate as a Simple Query [page 38]
Add an Aggregate simple query to the sample diagram to create a volume weighted average price (VWAP).

Creating a Join as a Simple Query [page 40]
Add a join to the sample project. A join combines events from two or more inputs to create a single stream or window. It is similar to a join in SQL.

Completing the Sample Project [page 46]
Clean up the diagram by removing unused elements.

3.2 Creating the Sample Project

Define a new set of processing instructions for event data.

Prerequisites

Start the SAP Event Stream Processor Studio.

Procedure

1. Create a project:
   - In SAP HANA studio, go to `File > New > Project...`, then select `SAP Event Stream Processor > New SAP Project` or:
   - In ESP, go to `File > New > ESP Project...`.
   - In the SAP ESP Authoring perspective in SAP HANA studio, `File > New > Project...` then select `New Streaming Project`

2. For the purposes of this tutorial, in the Name field, enter `my_portfolio_valuation`.
   A valid project name:
   - Must start with a lowercase letter, underscore, or dollar sign.
   - All other characters must be lowercase letters, numbers, underscores, or dollar signs.
   - Must not contain spaces.
For your own projects you can use any name. To ensure that you can run the sample project you are creating, use the values listed here.

3. In the Directory field, accept the default location or browse to a directory in which to store the new project folder.
   ESP creates three files in the named directory:
   
   `<project_name>.ccl` contains the CCL code.
   `<project_name>.cclnotation` contains the diagram that corresponds to the `.ccl` file.
   `<project_name>.ccr` contains the project configuration.

   For example, for a project directory named "trades," ESP creates a `trades.ccl`, `trades.cclnotation`, and `trades.ccr` file in the `trades` directory.

4. Click Finish to create the project files.
   The new project opens in the visual editor with one input stream, NEWSTREAM, and an inline schema ready for editing.

**Related Information**

**Launching Studio** [page 14]
Start SAP Event Stream Processor Studio using the desktop shortcut, Windows Start menu, or the command line. If you installed the ESP plugin for SAP HANA studio, you can access SAP Event Stream Processor functionality within the SAP HANA studio by opening the SAP ESP Run-Test and SAP ESP Authoring perspectives.

### 3.3 Editing a Project Diagram

Edit projects in the Visual editor by adding shapes from the Palette to the project diagram, connecting them, and completing the configuration of each shape.

**Procedure**

1. If the sample project diagram is not already open in the Visual editor, open it now:
   a. In SAP ESP Authoring perspective, from Project Explorer, open the sample project, `my_portfolio_valuation`.
   b. Navigate to the `.cclnotation` file in your project folder and double-click `my_portfoliovaluation.cclnotation`.
2. Click in the diagram to begin editing using the Palette.
Tip

To make the Visual editor window full-screen, double-click the `<name>Diagram` tab at the top. Double-click again to revert.

3. Select the input stream element NEWSTREAM that was added automatically when you created the project, right-click, and choose `Delete Element`.

To run the sample project with example data, you must delete this element from the project before compiling.

- `Delete Element` — removes the element from the project.
- `Delete from Diagram` — removes the element from the diagram, but retains it in the project. When you run the project, everything in the project runs, even elements that are not on the diagram.

4. (Optional) To toggle between the Visual editor and the CCL editor, choose `Switch to Text` or `Switch to Visual` (F6).

Note

The Visual editor, like other graphical user interfaces, offers several ways to accomplish most tasks, although this guide may not list all of them. For example, in many contexts you can carry out an action by:

- Clicking a button or other icon in a shape, or on the main toolbar
- Using a shortcut key
- Double-clicking an element to open it
- Right-clicking to select from the context menu
- Selecting from the main menu bar
- Editing element values in the Properties view

3.4 Adding an Input Adapter

Attach an adapter by inserting it in the diagram, connecting it to a stream or window, and setting properties.

Context

This tutorial shows you how to insert an adapter, enable it for schema discovery, then generate and attach the input window and its schema automatically. This is the best practice for creating a schema when using an adapter that supports schema discovery.

Alternatively, SAP Event Stream Processor allows you to create the stream or window and then attach an adapter. Use this method for adapters that do not support schema discovery, or where you want to explicitly create an inline schema for input streams or windows.
Procedure

1. Open the Input Adapters compartment in the Palette (to the right of the diagram) and locate the adapter you want.
   For this example, choose the [File/Hadoop Event XML Input] adapter, which reads data from an XML file.
2. Click the adapter in the Palette, then click in the diagram.
   The adapter shape is inserted but its border is red, indicating it is not complete, until you define its properties and attach it to a stream or window.
3. In the adapter shape toolbar hovering outside the shape, click Edit Properties.
4. (Optional) In the Adapter Properties dialog, change Name to identify your adapter.
5. Configure the adapter for schema discovery:
   Required properties are in red.

   a. Click in the Value column for Directory and click the Browse button ( ...).
   b. Click the Browse button in the Directory dialog to select the folder with the data files you want the adapter to read. Click OK.
   For this example, specify the absolute path to the sample data installed with the product.
   c. Enter a wildcard (*) in the Value column for File (in Directory). Click OK.
6. Click OK, then press Ctrl+S to save.

Next Steps

Import the schema and create a connected input stream or window with the same schema as the data file.
3.5 Discovering a Schema

Use the Schema Discovery button in the Adapter shape to discover and automatically create a schema based on the format of the data from the adapter.

Prerequisites

Add the adapter to the diagram and set its properties.

Procedure

1. Click Schema Discovery on the adapter toolbar. Studio displays a Progress Information box and looks for the configuration.
   - If the schema is configured properly and one or more data sets are found, a Schema Discovery: Select Schema dialog appears where you can view and select a schema.
   - If the schema is not successfully discovered, an error message appears stating that no schema was discovered for the adapter. You can:
     - Check that the adapter properties are configured for schema discovery.
     - Check the SAP Event Stream Processor: Studio Users Guide to see if the adapter supports schema discovery.

2. Select the schema you need. You can expand the data set to view the schema.
   - For this example, select positions.xml, then click Next.

3. In the Schema Discovery: Create Element dialog, choose Create new input window (with inline schema). This option creates and attaches a new window to the adapter, creates an inline schema for the window, and populates the window with the schema discovered from the adapter.
   - When the adapter is not yet attached to a stream or window, other options are:
     - Create a new input stream (with inline schema). Creates and attaches a new stream to the adapter, creates an inline schema for the stream, and populates the stream with the schema discovered from the adapter.
Create a new input stream (with attached schema).
Creates and attaches a new stream to the adapter, creates and attaches a new named schema to the stream, and populates the stream with the schema discovered from the adapter.

Create a new input window (with attached schema).
Creates and attaches a new window to the adapter, creates and attaches a new named schema to the window, and populates the window with the schema discovered from the adapter.

Create new named schema.
Creates a new named schema and populates it with the schema discovered from the adapter.

4. Click Finish.
   - The new input window appears with the default name positions_xml_window1, and is automatically connected to the File/Hadoop Event XML Input adapter.
   - The adapter file property is set. The red warning border disappears, indicating that the element is now valid.

5. In the Schema compartment of the input window, click the Toggle Key buttons for the BookId and Symbol columns to specify the primary key.
   - The button indicates primary key columns. With the primary key, the shape becomes valid.

6. Set a keep policy in the input window:
   a. Click Set Keep Policy.
   b. In the Edit Keep Policy dialog, choose All Rows and click OK.

7. Click the input window Edit button and name it positions. Be sure to use lower case.

Next Steps
Create another input window, PriceFeed. Either:

- Create the PriceFeed input window manually, following steps in the next task, or,
- Insert another File/Hadoop Event XML Input adapter and configure it for schema discovery. This time, when you discover the schema, choose pricefeed.xml in the exampledata directory. Name the input window PriceFeed, click the Id column to make it the primary key, and set the keep policy to 10 MIN.

Related Information

Schema Discovery Using Input Adapters [page 29]
In the tutorial you use the schema discovery feature to discover an external schema and create a CCL schema based on the format of the data from the datasource connected to an adapter.
3.6 Adding an Input Window Manually

Add an input window to the diagram in the sample `my_portfoliovaluation` project.

**Context**

These steps let you create an input window directly, and define the schema, without importing a schema.

If you used the input adapter to discover the schema and generated both input windows automatically, skip these steps and go directly to the next task.

**Procedure**

1. In the Visual editor, in the Palette to the right of the diagram, open the `Streams and Windows` compartment.
2. Click `Input Window`.
3. Click in an empty area in the diagram where you want to insert the input window.
   The input window object is added to the project. The red border indicates that it needs more definition to be valid.
4. To set the name of the input window, either:
   - In iconic mode, click once to select the shape, then click again to edit the name.
   - In verbose mode, click the edit icon next to the name.
   For this example, enter the name `PriceFeed`.
5. Click the "plus" sign to expand the shape to verbose mode if necessary, and click `Add Column` on the toolbar in the input window, to add each new column.

<table>
<thead>
<tr>
<th>Tip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hover over any icon to see its name.</td>
</tr>
</tbody>
</table>

A new column is created with a default name, and default datatype of integer.

6. Specify additional columns.
   a. Double-click each column name to edit it.
   b. Then double-click each datatype to select the correct datatype.
   For this example, enter these column names and datatypes:

   - **Id** integer
   - **Symbol** string
   - **TradeTime** seconddate
   - **Price** float
7. Click the button for the Id column to toggle it to the Key symbol.
Input windows require a primary key.
The Id column is now the primary key for the PriceFeed input window. The red warning border disappears, indicating that the element is now valid.

8. Create a retention window.
   a. Click Set Keep Policy.
   b. In the Edit Keep Policy dialog, choose Time, and enter 10 MIN in the text box to its right. Click OK.
   The default policy is to keep all rows of incoming data.
   This step defines a CCL KEEP clause, and retains all price events received in the last 10 minutes. Without a KEEP clause, the PriceFeed window would grow infinitely large. For more information on specifying a retention policy, see the SAP Event Stream Processor: Studio Users Guide.

9. Save (Ctrl+S).
   This saves changes to both the .cclnotation file (the diagram) and the .ccl file (the CCL).

Results

The input window and its schema (or deduced schema) are in the diagram.

3.7 Creating an Aggregate as a Simple Query

Add an Aggregate simple query to the sample diagram to create a volume weighted average price (VWAP).

Context

An Aggregate query groups events that have common values, and computes summary statistics for the group.

Procedure

1. In the Visual editor Palette, in Streams and Windows, click Aggregate.
2. Click in the diagram to create the object.
3. Change the default name, Aggregate1, to VWAP.
4. Connect PriceFeed to the VWAP aggregate:
   a. Click the Connector tool in the Palette.
b. Click the `PriceFeed` input window, then click the `VWAP` aggregate. Click the shape that produces the output first, then the shape that receives the data, to indicate the direction of data flow. Watch for visual indicators that show you when the connection is valid.

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Arrow]</td>
<td>Connection is allowed</td>
</tr>
<tr>
<td>![Arrow]</td>
<td>Connection is not allowed</td>
</tr>
</tbody>
</table>

5. Enter Column Expressions:
   a. Click `Add Column Expressions` ( ), then `Copy Columns from Input` ( ) in the shape toolbar to select the columns to copy into the schema for the aggregate window.
      For this example, copy these columns:
      - `PriceFeed.Symbol`
      - `PriceFeed.TradeTime`
      - `PriceFeed.Price`
   b. Edit column names to clarify that these columns will hold the most recent price and time for the group:
      - Change `TradeTime` to `LastTime`
      - Change `Price` to `LastPrice`
   c. Add additional columns by clicking `Add Column Expression` ( ) in the shape toolbar.
      For this example, add another column and edit its name to `VWAP`.

6. Edit a column expression by double-clicking to open the inline editor, by selecting one of the tabs in the Properties view, or by selecting an expression and pressing `Ctrl+F2` to edit it using the pop-up expression editor.
   For this example, edit the `VWAP` column expression to:
   ```
   sum ( PriceFeed.Price * PriceFeed.Shares ) / sum ( PriceFeed.Shares )
   ```

7. Click `Add Group Clause ( )` to edit the grouping of columns in the aggregate object.

   **Note**
   The Aggregate shape must have exactly one GROUP BY expression.

   For this example, select `PriceFeed.Symbol` as the grouping column.

   The red warning border disappears, indicating that the element is now valid.

8. Set a keep policy in the input window:
   a. Click `Set Keep Policy` 
   b. In the Edit Keep Policy dialog, choose `All Rows` and click `OK`. 
Results

By default, the Aggregate is created as Output, which allows external applications to subscribe to or query it, and allows you to view it using the Stream View in the SAP ESP Run-Test perspective.

Related Information

Simple Queries [page 30]
Accomplish most common querying tasks using a set of simple queries available in the Visual editor: filter, aggregate, join, compute, union, and pattern.

3.8 Creating a Join as a Simple Query

Add a join to the sample project. A join combines events from two or more inputs to create a single stream or window. It is similar to a join in SQL.

Context

Event Stream Processor supports inner joins, left and right outer joins, and full outer joins, with join syntax comparable to SQL ANSI join syntax and comma-separated syntax. For more information about joins, see the SAP Event Stream Processor: Studio Users Guide or the SAP Event Stream Processor: Developer Guide.

Procedure

1. In the Visual editor Palette, in Streams and Windows, select Join.
   If necessary, close the compartments below Streams and Windows, or use the arrow below the compartment, so that Join is visible.
2. Click in the diagram to create the object.
   For this example, edit the join object name to be IndividualPositions.
3. Using the Connector tool, connect the join object to the appropriate stream or window.
   Attach join objects to any stream, window, or Flex operator. Join objects have multiple inputs, but only one output.

   Note
   Streams, windows, keyed streams, and delta streams can participate in a join. But, only one stream can participate in a join. A delta stream may participate in a join only if it has a KEEP clause specified. And, SAP
recommends that only expert users use delta streams: they cannot be added to a project using Studio, they can only be defined in CCL.

For this example, connect the VWAP aggregate object and the Positions input window to the IndividualPositions join object, in that order.

**Tip**

To add multiple connections, Shift+click and hold the Connector tool and add connections. To return to normal selection, press Esc or click the Select tool in the Palette to release it.

4. Click Add Column Expressions \( \text{Add Column Expressions} \), then Copy Columns \( \text{Copy Columns} \) in the join shape toolbar and select columns to copy.

**Tip**

If you get an error, or do not see all columns from both inputs listed, try reconnecting the new Join element to the Positions or VWAP shapes as needed.

For this example, choose Select All, then clear the check box on VWAP.Symbol so that you don't get the symbol field twice.

5. Click Add Column Expressions \( \text{Add Column Expressions} \).

For this example add two columns: CurrentPosition and AveragePosition.

6. To modify column expressions, either:
   - Double-click on the expression to open the inline editor, and either type directly or press Ctrl+Space for syntax completion assistance, to pick from column names and functions, or.
   - Press Ctrl+F2 to open the expression editor. Press Ctrl+Space to display the available input columns and built-in functions, or enter the desired expression manually, or.
   - Modify the expression in the Properties view.

For this example, create these Column Expressions:

- CurrentPosition: \( (\text{VWAP.LastPrice} \times \text{positions.SharesHeld}) \)
- AveragePosition: \( (\text{VWAP.VWAP} \times \text{positions.SharesHeld}) \)

7. In the Join Conditions compartment of the join shape, set up the join conditions.

If you connected the join to the VWAP and Positions inputs, in that order, there are now two elements in the Join Conditions compartment. The first defines the leftmost element for the join. If you connected to VWAP first, the first element (left side of the join) is VWAP. For this example, you must configure the second join element.

   a. Double-click the second join element to open the Edit Join Expression dialog.
   b. Choose a join type.
      - For this example, use RIGHT, which is a right outer join. You want RIGHT because VWAP is the first, or left input, and Positions is the second, or right input. You only want your positions in the output; you do not need prices for symbols that are not held in the portfolio.
   c. Select the columns to join on.
      - You cannot edit join constraints manually in the Visual editor.
      - For this example:
        - As Source 1, ensure that VWAP is in the dropdown, and select the option starting with Symbol: as the column.
○ As Source 2, ensure that Positions is in the dropdown, and select the option starting with Symbol: as the column.

```
Select the type of join and the data source to add to the join.

Options          Source to join
Left             Positions
Full
Right
Inner

Build an ON clause by selecting one column from Source 1 and one column from Source 2.

Source 1          Source 2
VWAP             Positions
LastPrice : float
LastTime : date
Symbol : string
VWAP : function
BookId : string
SharesHeld : integer
Symbol : string
```

d. Click Add. If a message displays indicating that datatypes for the selected columns should match, click Yes to continue.

The columns chosen appear in Join Constraints, where you should now see:

```
ON VWAP.Symbol = positions.Symbol
```

The dialog shows:

e. Click OK.

8. Set a keep policy in the input window:
9. In the join shape, click (Toggle Type to OUTPUT), if this option is available. (If it is not, the shape is already set to OUTPUT).

Results

The IndividualPositions join shape now shows the completed join, as shown in the figure.

Related Information

Simple Queries [page 30]

Accomplish most common querying tasks using a set of simple queries available in the Visual editor: filter, aggregate, join, compute, union, and pattern.
3.9 Preparing Data for Output

Add a new aggregate to finalize the data and prepare it for output into SAP HANA.

Context

If you do not have SAP HANA installed, complete these steps, then move on to Completing the Sample Project [page 46].

Procedure

1. Create an additional Aggregate Simple Query and name it ValueByBook.
   a. Connect the IndividualPositions join object to it.
   b. Click Copy Columns in the shape toolbar and copy columns BookId, CurrentPosition, and AveragePosition.
   c. Set column expressions:
      ○ BookId IndividualPositions.BookId
      ○ AveragePosition sum ( IndividualPositions.AveragePosition )
   d. Add the GROUP BY clause ( { } ) IndividualPositions.BookId
   e. Toggle to OUTPUT if the shape is not set as OUTPUT already.

   Tip

   Use the inline editor. Double-click on the column expression, and use the Home and End keys to quickly edit the expression.

Next Steps

Set up a connection and add an output adapter to deposit data into an SAP HANA Studio table. If you do not have SAP HANA installed, skip to Completing the Sample Project [page 46].
3.10 Adding an Output Adapter for SAP HANA

Create a service definition for an ODBC connection to an SAP HANA database, and insert and configure a HANA Output adapter.

Context

If you do not have SAP HANA installed, skip this task and move on to Completing the Sample Project [page 46].

Procedure

1. In the Data Services View, right-click the server on which you want to create a new database service definition and select Add ODBC Service.
2. Select the new database service. In the Properties view, set the Name parameter to hanaservice. A database service name must be:
   ○ unique
   ○ case-sensitive
   ○ must begin with a letter
   ○ may contain a character string consisting of either letters, numbers, underscores, dots, and colons.
   This service name is the value you specify to components, such as the SAP ESP Authoring HANA Output adapter, that access external databases.
3. From the Driver drop down list, select streamingdbodbc64_lib or streamingdbodbc_lib.
4. Set the User parameter to the user name that you want to use when communicating with SAP HANA.
5. Set the Password parameter to the password for your user name.
6. Select Enable as HANA Reference Service to set this database service as a HANA service entry.
7. Click anywhere outside the Properties view to save your work.
8. Open the Output Adapters compartment in the Palette and choose HANA Output. This adapter uses multiple parallel ODBC connections to load information from ESP into the SAP HANA server.
   a. Add the adapter to the diagram and name it toSAPHANA.
   b. Click Edit Properties ( ). Property values in red must be filled in.
      ○ Set Database Service Name to hanaservices.
      ○ Set Target Database Table Name to the required SAP HANA table. In this example, you will not be configuring the project from the SAP HANA side, so enter exampletable in the value field.
   c. Open the Advanced tab:
      ○ Set Data Warehouse Mode to ON.
      ○ In the MSDate value field, enter TIME.
   d. Click OK, then press Ctrl+S to save
3.11 Completing the Sample Project

Clean up the diagram by removing unused elements.

Procedure

1. Delete any unused elements from the project so that you can run it.
   For example, if you have not done so, remove the unused input stream element NEWSTREAM that was added automatically when you created the project.

2. (Optional) Toggle to Iconic mode or Verbose mode.
   ○ Click the Toggle Image button in the upper left corner of a shape, or,
   ○ Click the All Iconic or All Verbose button in the toolbar.

3. (Optional) Click Layout Left to Right to line up shapes.

4. (Optional) To close the diagram, press Ctrl+W or Ctrl+F4, or click the X on the tab at the top of the editor.

Results

The completed diagram should look like this in Verbose mode. You might need to open some compartments and click again to see details for all elements.

![Completed Sample Portfolio Valuation Diagram](image)

Figure 5: Completed Sample Portfolio Valuation Diagram

Next Steps

Follow the procedures Tutorial: Testing a Project, to compile and test the sample project in Studio, using test data provided in your installation.
Related Information

Simple Queries [page 30]
Accomplish most common querying tasks using a set of simple queries available in the Visual editor: filter, aggregate, join, compute, union, and pattern.
4 Tutorial: Testing a Project

Compile, run, and test the project you have created by using tools in the SAP ESP Run-Test perspective.

1. **Compiling the Sample Project** [page 48]
   Compile a project before running it to check for errors and make corrections.

2. **Viewing Problems** [page 49]
   Use the Problems view to view error details when trying to validate, upload, and compile projects.

3. **Deploying the Sample Project** [page 50]
   Run the project and watch it open in the SAP ESP Run-Test perspective.

4. **Loading Data into the Sample Project** [page 51]
   Test the sample project by loading reference data into the Positions window.

5. **Testing the Project with Recorded Data** [page 52]
   Play back the previously recorded price feed data, and view the continuous portfolio valuations in the sample project.

6. **Other Tools for Running and Testing Projects** [page 53]
   SAP Event Stream Processor Studio includes other tools for testing projects.

Related Information

**Project Execution and Testing** [page 25]
Run and test all aspects of a project using SAP Event Stream Processor Studio.

### 4.1 Compiling the Sample Project

Compile a project before running it to check for errors and make corrections.

**Procedure**

1. If the sample project is not already open in the Visual editor, open it now.
   a. Go to SAP ESP Authoring perspective.
   b. In Project Explorer, expand the *my_portfolio_valuation* folder.
   c. Double-click *my_portfolio_valuation.cclnotation*.

2. To compile the project, either:
   ○ Click the **Compile Project** button in the main toolbar, or,
   ○ Press **F7**.
Results

The project compiles and reports any errors found. Compilation errors are displayed in the Problems or Console view, depending on the type of error.

Next Steps

Review and resolve any problems. If it compiles with no errors, you can skip Viewing Problems.

4.2 Viewing Problems

Use the Problems view to view error details when trying to validate, upload, and compile projects.

Prerequisites

Open the SAP ESP Authoring perspective.

Procedure

1. Click on a problem in Problems view, or expand the group to see individual errors. By default, Problems view is at the bottom of the screen, and problems are grouped by severity. Error details appear in Problems view and in the status bar at the bottom left side of the screen.

   ➤ Tip
   If you double-click on a problem in the problems view while the project is open in the Visual editor, the CCL editor opens read-only to show you where the problem is. To fix the problem, either:
   ○ Return to the Visual editor and fix it there, or,
   ○ Close both the Visual editor and CCL editor for the project, and then reopen the project in the CCL editor.

2. If the error message is too long to show the entire message, click it to read the full text in the status bar at the bottom of the Studio window.

3. Right-click an item to choose from the context menu:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Go to</td>
<td>Highlight the problem in the .ccl file. The CCL editor opens in read-only mode.</td>
</tr>
<tr>
<td>Option</td>
<td>Action</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Copy</td>
<td>Copy error details to the clipboard. When you exit Studio, the contents of problems view are removed. Use this option to save off errors.</td>
</tr>
<tr>
<td>Show in</td>
<td>Display details in Properties view.</td>
</tr>
<tr>
<td>Quick Fix</td>
<td>(Disabled)</td>
</tr>
<tr>
<td>Properties</td>
<td>Display details in a dialog box.</td>
</tr>
</tbody>
</table>

4. (Optional) Click the View menu dropdown to see more options.
5. Click the Console tab to view compiler results.

4.3 Deploying the Sample Project

Run the project and watch it open in the SAP ESP Run-Test perspective.

Prerequisites

Make sure the project compiles without errors. Correct any problems before you can run the project.

Procedure

1. With the diagram open in the editor, click 🔄 in the main toolbar.

   **Note**

   If you are prompted for a user name and password for the local cluster, you must first change the default system-generated credentials. See *SAP Event Stream Processor: Studio Users Guide > Troubleshooting > Prompted for Local Cluster Password*.

2. Review the running project in SAP ESP Run-Test perspective.
4.4 Loading Data into the Sample Project

Test the sample project by loading reference data into the Positions window.

Context

If your project has the File/Hadoop Event XML Input adapter attached to the Positions input window, data is loaded automatically when you start the project. If you removed or omitted the adapter, use this alternative process to load the sample data.

Procedure

1. In Server View, expand the `my_portfolio_valuation` project to show the list of windows and streams in the project.
2. Double-click the Positions input window to open it in Stream View. Stream View is in the upper right-hand portion of SAP ESP Run-Test perspective.
   ○ If your diagram has the File/Hadoop Event XML Input adapter connected to the Positions input window, Stream View shows sample data for Positions, loaded automatically from the adapter.
   ○ If you removed the adapter, go to the next step to load the data manually.
3. Load positions data from a file into the Positions window.
   a. Go to File Upload view.
   b. Click the Select Project button in the view toolbar. If you only have one project open in Server View, no dialog appears and ESP automatically selects the project for you. Otherwise, select the `my_portfolio_valuation` project in the dialog, and click OK.
   c. Click the Browse button, navigate to your \SybaseESP\5.1\workspace\exampledata folder, and select positions.xml. If you do not see the positions.xml file, try changing the file name extension filter in the lower right corner of the dialog to * .xml.
   d. Click Open.
   e. With positions.xml highlighted in File Upload view, click the Upload button. Watch the data flow in Stream View, as Studio loads the three positions for Book1 and Book2.
4.5 Testing the Project with Recorded Data

Play back the previously recorded price feed data, and view the continuous portfolio valuations in the sample project.

Procedure

1. In Server View, double-click the IndividualPositions, VWAP, and ValueByBook output windows.
   In the Server View list, a red arrow in the lower right corner of the window icon (□) indicates the output windows.
2. Click the Playback tab.
3. If necessary, click the Select Project (□) button in the upper right corner of Playback view.
   ○ If you only have one project running, Studio selects it for you.
   ○ Otherwise, select the my_portfolio_valuation project in the dialog and click OK.
4. Click the Select Playback File (□) button.
5. Navigate to your <ESP-workspace-folder> \ exampledata folder, and select pricefeed.xml. Click Open.
   If you do not see the pricefeed.xml file, change the file name extension filter to * .xml.
6. In Playback view, in the Playback Mode frame, click the rec/ms button, then enter a rec/ms value of 1.
   A value of 1 plays back at a rate of 1000 records per second.
7. Click the green Start Playback (□) button to start playback of the price feed.
8. While the data plays back, click each of the output windows in Stream View to see the calculations revised in real-time.
9. (Optional) Click Event Tracer view, choose Select Running Project, and click Initialize with Base Data.
   In this example, Event Tracer shows the PriceFeed and Positions elements in green to indicate Insert operations. VWAP, IndividualPositions, and ValueByBook are in blue, indicating Updates. Colors change as different event types are processed.
   Double-click each node to watch event data in the Console.
10. To stop the playback, click Stop.

11. When you are done testing the project, right-click it in Server View and choose Stop Project. If you omit this step, the project stops when you exit Studio, but you may get an error.

Tip

If you see an error when you restart Studio, or when you try to open a .ccl file after running a project, there may be multiple instances of Studio trying to use the same Studio workspace location. If this occurs, close Studio and restart it.

4.6 Other Tools for Running and Testing Projects

SAP Event Stream Processor Studio includes other tools for testing projects.

For information beyond the scope of this guide on running, configuring, monitoring, querying, and debugging projects, see the SAP Event Stream Processor: Studio Users Guide.

Related Information

The Run-Test Perspective [page 19]:

In the SAP ESP Run-Test perspective, you access tools to test, monitor, debug, and fine-tune a project.
5 Continuous Computation Language

CCL is the primary event processing language of SAP Event Stream Processor. Projects are defined in CCL.

CCL is based on Structured Query Language (SQL), adapted for stream processing.

CCL supports sophisticated data selection and calculation capabilities, including features such as data grouping, aggregations, and joins. However, CCL also includes features that are required to manipulate data during real-time continuous processing, such as windows on data streams, and pattern and event matching.

The key distinguishing feature of CCL is its ability to continuously process dynamic data. A SQL query typically executes only once each time it is submitted to a database server and must be resubmitted every time a user or an application needs to reexecute the query. By contrast, a CCL query is continuous. Once it is defined in the project, it is registered for continuous execution and stays active indefinitely. When the project is running in the ESP server, a registered query executes each time an event arrives from one of its datasources.

Although CCL borrows SQL syntax to define continuous queries, the ESP server does not use a SQL query engine. Instead, it compiles CCL into a highly efficient byte code that is used by the ESP server to construct the continuous queries within the data-flow architecture.

CCL queries are converted to an executable form by the CCL compiler. ESP servers are optimized for incremental processing, hence the query optimization is different than for databases. Compilation is typically performed within Studio, but it can also be performed by invoking the CCL compiler from the command line.

5.1 CCLScript

CCLScript is a scripting language that brings extensibility to CCL, allowing you to create custom operators and functions that go beyond standard SQL.

The ability to embed CCLScript scripts in CCL provides tremendous flexibility, and the ability to do it within the CCL editor maximizes user productivity. CCLScript also allows you to define any complex computations that are easier to define using procedural logic rather than a relational paradigm.

CCLScript is a simple scripting language comprised of expressions used to compute values from other values, as well as variables, and looping constructs, with the ability to organize instructions in functions. CCLScript syntax is similar to C and Java, though it also has similarities to languages that solve relatively small programming problems, such as AWK or Perl.

Related Information

CCL Authoring [page 55]

The CCL editor is a text authoring environment within SAP Event Stream Processor Studio where you can edit CCL code.

Editing in the CCL Editor [page 56]

Update and edit CCL code as text.

CCL for the Sample Project [page 57]
The CCL for the Portfolio Valuation sample project created in the Studio Visual editor is shown here, with the corresponding shape in the diagram for each element. Line breaks are added for readability.

CCL for Sample Project with Modules [page 60]
This variation of the portfolio valuation project uses a defined module with a named schema to easily scale out the application in a very high volume deployment. The SAP HANA Output adapter is not included in this module.

5.2 CCL Authoring

The CCL editor is a text authoring environment within SAP Event Stream Processor Studio where you can edit CCL code.

You can work in the CCL editor exclusively, or use it as a supplement to the Visual editor. The CCL editor offers syntax completion options, syntax checking, and error validation.

A single CCL file can be open in only one editor at a time. The Visual and CCL editors are completely integrated: when you save and switch to the other editor, your work is saved there as well.

Most users new to Event Stream Processor find it easier to get started in the Visual editor. As you gain experience with the product, and learn to successfully compile and run a simple project, you may want to use the CCL editor to add advanced features to your projects.

The SAP Event Stream Processor: Studio Users Guide explains use of the CCL editor within Studio.

For CCL language usage and reference details, see the SAP Event Stream Processor: Developer Guide.

Related Information

CCLScript [page 54]
CCLScript is a scripting language that brings extensibility to CCL, allowing you to create custom operators and functions that go beyond standard SQL.

Editing in the CCL Editor [page 56]
Update and edit CCL code as text.

CCL for the Sample Project [page 57]
The CCL for the Portfolio Valuation sample project created in the Studio Visual editor is shown here, with the corresponding shape in the diagram for each element. Line breaks are added for readability.

CCL for Sample Project with Modules [page 60]
This variation of the portfolio valuation project uses a defined module with a named schema to easily scale out the application in a very high volume deployment. The SAP HANA Output adapter is not included in this module.
5.3 Editing in the CCL Editor

Update and edit CCL code as text.

**Procedure**

1. Select the SAP ESP Authoring perspective.
2. In Project Explorer, expand the project container, and double-click the `.ccl` file name to open it in the CCL editor.
   
   **Note**
   Advanced CCL users can include multiple CCL files in the same project, by using an IMPORT statement to import shared schemas and module definitions from another file.

3. Begin editing text in the CCL editor window.

   **Tip**
   If you open a `.ccl` file in the CCL editor when the same project is open in the Visual editor, the CCL editor opens in read-only mode and you cannot edit the file.
   Close both the Visual editor and CCL editor for the project, and then reopen the project in the CCL editor.

   **Note**
   Backslashes within string literals are used as escape characters. Any Windows directory paths must therefore be specified with two backslashes.

4. (Optional) Press `Ctrl+Space` to show a syntax completion proposal.
5. (Optional) To insert CREATE statement template code, right-click, choose *Create*, and then choose the element to create.
6. Choose **File** ➔ **Save** (`Ctrl+S`) to save the `.ccl` file and the project.

**Related Information**

*CCLScript* [page 54]
*CCLScript is a scripting language that brings extensibility to CCL, allowing you to create custom operators and functions that go beyond standard SQL.*

*CCL Authoring* [page 55]
*The CCL editor is a text authoring environment within SAP Event Stream Processor Studio where you can edit CCL code.*

*CCL for the Sample Project* [page 57]
CCL for Sample Project with Modules [page 60]

This variation of the portfolio valuation project uses a defined module with a named schema to easily scale out the application in a very high volume deployment. The SAP HANA Output adapter is not included in this module.

5.4 CCL for the Sample Project

The CCL for the Portfolio Valuation sample project created in the Studio Visual editor is shown here, with the corresponding shape in the diagram for each element. Line breaks are added for readability.

<table>
<thead>
<tr>
<th>CCL</th>
<th>Diagram</th>
</tr>
</thead>
</table>
| CREATE INPUT WINDOW PriceFeed  
SCHEMA  
( Id integer ,  
Symbol string ,  
TradeTime seconddate ,  
Price float ,  
Shares integer )  
PRIMARY KEY ( Id )  
KEEP 10 MIN ; | ![Diagram of PriceFeed window] |

Table 3: Portfolio Valuation Project CCL and Diagram Elements
```ccl
/**@SIMPLEQUERY=AGGREGATE*/
CREATE OUTPUT WINDOW VWAP
PRIMARY KEY DEDUCED
KEEP ALL ROWS
AS
SELECT
  PriceFeed.Symbol Symbol,
  PriceFeed.TradeTime LastTime,
  PriceFeed.Price LastPrice,
  sum ( PriceFeed.Price * PriceFeed.Shares ) / sum ( PriceFeed.Shares ) VWAP
FROM PriceFeed
GROUP BY PriceFeed.Symbol;
```

```ccl
CREATE INPUT WINDOW Positions
SCHEMA
  ( BookId string,
    Symbol string,
    SharesHeld integer )
PRIMARY KEY ( BookId, Symbol ) KEEP ALL ROWS;
```
```sql
/**@SIMPLEQUERY=JOIN*/
CREATE OUTPUT WINDOW IndividualPositions
PRIMARY KEY DEDUCED
KEEP ALL ROWS
AS
SELECT
  VWAP.LastTime LastTime ,
  VWAP.LastPrice LastPrice ,
  VWAP.VWAP VWAP ,
  Positions.BookId BookId ,
  Positions.Symbol Symbol ,
  Positions.SharesHeld SharesHeld ,
  VWAP.LastPrice * Positions.SharesHeld CurrentPosition ,
  VWAP.VWAP * Positions.SharesHeld AveragePosition
FROM VWAP RIGHT JOIN Positions ON
  VWAP.Symbol = Positions.Symbol ;

/**@SIMPLEQUERY=AGGREGATE*/
CREATE OUTPUT WINDOW ValueByBook
PRIMARY KEY DEDUCED
KEEP ALL ROWS
AS
SELECT
  IndividualPositions.BookId BookId ,
  sum ( IndividualPositions.AveragePosition ) AveragePosition
FROM IndividualPositions
GROUP BY IndividualPositions.BookId ;

ATTACH INPUT ADAPTER Adapter1
TYPE toolkit_file_xmllist_input
TO
Positions
PROPERTIES dir = 'C:/Documents and Settings/\<username>/My Documents/SybaseESP/5.1/workspace/exampledata' ,
file = 'positions.xml' ;
```
CCL

<table>
<thead>
<tr>
<th>CCL</th>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTACH OUTPUT ADAPTER toSAPhana TYPE hana_out</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>TO ValueByBook</td>
<td></td>
</tr>
<tr>
<td>PROPERTIES service = 'hanaservice',</td>
<td></td>
</tr>
<tr>
<td>table = 'exampletable',</td>
<td></td>
</tr>
<tr>
<td>dataWarehouseMode = 'ON',</td>
<td></td>
</tr>
<tr>
<td>msdateColumnName = 'TIME';</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Related Information

**CCLScript [page 54]**

CCLScript is a scripting language that brings extensibility to CCL, allowing you to create custom operators and functions that go beyond standard SQL.

**CCL Authoring [page 55]**

The CCL editor is a text authoring environment within SAP Event Stream Processor Studio where you can edit CCL code.

**Editing in the CCL Editor [page 56]**

Update and edit CCL code as text.

**CCL for Sample Project with Modules [page 60]**

This variation of the portfolio valuation project uses a defined module with a named schema to easily scale out the application in a very high volume deployment. The SAP HANA Output adapter is not included in this module.

5.5  **CCL for Sample Project with Modules**

This variation of the portfolio valuation project uses a defined module with a named schema to easily scale out the application in a very high volume deployment. The SAP HANA Output adapter is not included in this module.

The module, valuation.ccl, computes the VWAP aggregate, and does the join to the Positions window. The project uses the module to divide the moving data into smaller partitions, based on the first letter of the Symbol column. This strategy spreads the load out to more cores, thereby increasing throughput. By using modules, with very little coding you can easily double, quadruple, and so on, the number of partitions.

This example also implements the streaming tick data in PriceFeed as a stream rather than an input window. Because keeping every tick would use a lot of memory, and because the state is never updated or queried, a stream is a more likely choice than a window in a real-world scenario for this event stream.
Example

Create Module valuation

The valuation module:

1. Defines the input stream TradesIn.
2. Defines a stream, Filter1, that filters TradesIn data into a substream based on the declared parameters `afrom` and `ato`.
3. Defines the input window Portfolio.
4. Defines the VWAP aggregate as an output window.
5. Defines another output window, ValueBySymbol, that performs a join similar to the Join simple query in the simple PortfolioValuation project, with the addition of a cast for the float data.

```sql
CREATE MODULE valuation
IN TradesIn, Portfolio
OUT ValueBySymbol, VWAP
BEGIN
IMPORT 'import.ccl';
DECLARE
PARAMETER STRING afrom;
PARAMETER STRING ato;
END;
CREATE INPUT STREAM TradesIn
SCHEMA TradesSchema;
CREATE STREAM Filter1 AS
SELECT * FROM TradesIn
WHERE substr(TradesIn.Symbol,1,1) >= afrom
    and substr(TradesIn.Symbol,1,1) <= ato;
CREATE INPUT WINDOW Portfolio
SCHEMA PortfolioSchema;
CREATE OUTPUT WINDOW VWAP
PRIMARY KEY DEDUCED AS
    SELECT Filter1.Symbol Symbol,
        (sum((Filter1.Price * cast(FLOAT ,Filter1.Shares))) / cast(FLOAT ,sum(Filter1.Shares)))
        AS VWAP,
        sum (Filter1.Shares ) Total_Shares,
        valueinserted(Filter1.Price) LastPrice,
        valueinserted(Filter1.TradeTime) TradeTime
    FROM Filter1
    GROUP BY Filter1.Symbol;
CREATE OUTPUT WINDOW ValueBySymbol
SCHEMA (BookId STRING, Symbol STRING, CurrentPosition FLOAT,
AveragePosition FLOAT);
PRIMARY KEY (BookId, Symbol) AS
SELECT
    Portfolio.BookId AS BookId,
    Portfolio.Symbol AS Symbol,
    (VWAP.LastPrice * cast(FLOAT ,Portfolio.SharesHeld))
        AS CurrentPosition,
    (VWAP.VWAP * cast(FLOAT ,Portfolio.SharesHeld))
        AS AveragePosition
FROM Portfolio JOIN VWAP
ON Portfolio.Symbol = VWAP.Symbol;
```
Example

Create Named Schema TradesSchema

```sql
CREATE SCHEMA TradesSchema
  ( Id integer ,
  Symbol string ,
  TradeTime second date ,
  Price float ,
  Shares integer ) ;
```

Example

Create Named Schema PortfolioSchema

```sql
CREATE SCHEMA PortfolioSchema
  ( BookId string ,
  Symbol string ,
  SharesHeld integer ) ;
```

Example

Import and Load the valuation Module

In the parent scope, the valuation module is loaded three times, as Valuation1, Valuation2, and Valuation3.

1. The IN clause binds the input streams in the module to streams in the parent scope. TradesIn is bound to InputStream1, and Portfolio is bound to InputPositions.
2. The OUT clause binds the output window in the module, ValueBySymbol, with the three parameterized output windows, VbySym1, VbySym2, and VbySym3, and partitions the VWAP aggregate as VWAP1, VWAP2, and VWAP3.

```sql
IMPORT 'import.ccl';
IMPORT 'valuation.ccl';
DECLARE
  PARAMETER STRING afrom := 'A';
  PARAMETER STRING ato := 'Z';
END;

CREATE INPUT STREAM InputStream1 SCHEMA TradesSchema ;
CREATE INPUT WINDOW InputPositions SCHEMA PortfolioSchema PRIMARY KEY ( BookId , Symbol ) ;
LOAD MODULE valuation as Valuation1
  in TradesIn = InputStream1, Portfolio = InputPositions
  OUT ValueBySymbol = VbySym1, VWAP = VWAP1
  PARAMETERS afrom = 'A', ato = 'J'
; LOAD MODULE valuation as Valuation2
```
in TradesIn = InputStream1, Portfolio = InputPositions
OUT ValueBySymbol = VbySym2, VWAP = VWAP2
PARAMETERS afrom = 'K', ato = 'Q'
;
LOAD MODULE valuation as Valuation3
in TradesIn = InputStream1, Portfolio = InputPositions
OUT ValueBySymbol = VbySym3, VWAP = VWAP3
PARAMETERS afrom = 'R', ato = 'Z'
;
CREATE OUTPUT WINDOW UnionVWAP
PRIMARY KEY DEDUCED
AS SELECT * FROM VWAP1
UNION SELECT * FROM VWAP3
UNION SELECT * FROM VWAP2
;
CREATE OUTPUT WINDOW ValueBySymbol
PRIMARY KEY (BookId,Symbol)
AS SELECT * FROM VbySym1
UNION SELECT * FROM VbySym3
UNION SELECT * FROM VbySym2
;

// ----------------------------
// stream ValueByBook
CREATE OUTPUT WINDOW ValueByBook
SCHEMA (BookId STRING, CurrentPosition FLOAT, AveragePosition FLOAT)
PRIMARY KEY DEDUCED AS
SELECT ValueBySymbol.BookId AS BookId,
sum(ValueBySymbol.CurrentPosition) AS CurrentPosition,
sum(ValueBySymbol.AveragePosition) AS AveragePosition
FROM ValueBySymbol
GROUP BY ValueBySymbol.BookId;

ATTACH INPUT ADAPTER Adapter1 TYPE toolkit_file_xmllist_input TO InputStream1
GROUP nostartGroup
PROPERTIES
dir = '../exampledata' ,
file = 'pricefeed.xml'
, pollingPeriod = 0 ,
xmllistSecondDateFormat = 'yyyy-MM-ddTHH:mm:ss' ,
xmllistMsDateFormat = 'yyyy-MM-ddTHH:mm:ss';

ATTACH INPUT ADAPTER Adapter2 TYPE toolkit_file_xmllist_input TO InputPositions
PROPERTIES
dir = '../exampledata' ,
file = 'positions.xml'
, pollingPeriod = 0 ,
xmllistSecondDateFormat = 'yyyy-MM-ddTHH:mm:ss' ,
xmllistMsDateFormat = 'yyyy-MM-ddTHH:mm:ss';

ADAPTER START GROUPS nostartGroup nostart
;

Note
When sandboxing is enabled, edit the dir property. For example, change dir='../exampledata' to
dir='<sandbox-base-directory>/exampledata'. Otherwise, when sandboxing is enabled, the
example does not compile properly and you receive an error message.
Related Information

CCLScript [page 54]
CCLScript is a scripting language that brings extensibility to CCL, allowing you to create custom operators and functions that go beyond standard SQL.

CCL Authoring [page 55]
The CCL editor is a text authoring environment within SAP Event Stream Processor Studio where you can edit CCL code.

Editing in the CCL Editor [page 56]
Update and edit CCL code as text.

CCL for the Sample Project [page 57]
The CCL for the Portfolio Valuation sample project created in the Studio Visual editor is shown here, with the corresponding shape in the diagram for each element. Line breaks are added for readability.
Important Disclaimers and Legal Information

Coding Samples

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, unless damages were caused by SAP intentionally or by SAP’s gross negligence.

Accessibility

The information contained in the SAP documentation represents SAP’s current view of accessibility criteria as of the date of publication; it is in no way intended to be a binding guideline on how to ensure accessibility of software products. SAP in particular disclaims any liability in relation to this document. This disclaimer, however, does not apply in cases of wilful misconduct or gross negligence of SAP. Furthermore, this document does not result in any direct or indirect contractual obligations of SAP.

Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with "you", or a gender-neutral noun (such as "sales person" or "working days") is used. If when referring to members of both sexes, however, the third-person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.

Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. SAP does not warrant the availability and correctness of this related information or the ability of this information to serve a particular purpose. SAP shall not be liable for any damages caused by the use of related information unless damages have been caused by SAP’s gross negligence or wilful misconduct. All links are categorized for transparency (see: http://help.sap.com/disclaimer).