SAP HANA smart data streaming: technical overview

Applies to:
SAP HANA Smart Data Streaming 1.0 SPS09

Summary
This paper provides a technical overview of smart data streaming, a new capability of the SAP HANA Platform. It includes an introduction to event stream processing, how and where it’s used, explains how streaming projects are built and deployed, and then provides an overview of all the key features and how to use them.

Author:
Jeff Wootton, SAP HANA product management

Company: SAP

Created on: 21 November 2014

Author Bio
Jeff Wootton is currently a member of the SAP HANA product management team and the lead on smart data streaming. Jeff has over 20 years of experience in the field of real-time data distribution and management in event-driven architectures.
# Table of Contents

Introduction to Event Stream Processing ................................................................. 3  
  Event Streams ........................................................................................................... 3  
  Immediate Insight ..................................................................................................... 3  
  Complex Event Processing ......................................................................................... 4  
  Examples of Solutions using Event Stream Processing ........................................... 4  
  Basic Features of a Streaming Project ....................................................................... 5  
Smart Data Streaming: System Landscape ................................................................. 5  
  The Streaming Cluster ............................................................................................. 6  
Streaming Projects ..................................................................................................... 6  
Design Time Tools .................................................................................................... 8  
Interfaces and Adapters ............................................................................................ 9  
  Standard Adapters .................................................................................................. 9  
  Custom Adapters .................................................................................................... 9  
  Software Developers Kit ......................................................................................... 9  
Project Deployment ................................................................................................... 10  
CCL: Continuous Computation Language ............................................................... 10  
  Common CCL Query Types ....................................................................................... 10  
  The CCL Query graph – Data-flow Programming .................................................... 11  
  CCL Script for Versatility ......................................................................................... 11  
  CCL Script Controls and Data Structures ............................................................... 12  
  Flex operators .......................................................................................................... 12  
  Built-In function library ......................................................................................... 12  
  Modularity .............................................................................................................. 12  
System Administration ............................................................................................. 13  
Security .................................................................................................................... 13  
Resiliency ................................................................................................................. 13  
Guaranteed Data Delivery ........................................................................................ 13  
Scalability and Performance ..................................................................................... 14  
Unique Features ........................................................................................................ 15  
Related Content ....................................................................................................... 15  
Copyright .................................................................................................................. 16
Introduction to Event Stream Processing

SAP HANA smart data streaming is a high speed, highly scalable real-time event stream processor that provides the ability to process incoming information as fast as it arrives, capture useful information in the HANA database, and monitor the incoming data to enable you to respond immediately, as things happen.

Event Streams

In an increasingly wired world, event streams are all around us, continuously sending information about what's happening, as fast as things happen. Take just a few examples:

- Smart devices - sensor readings
- Scanners, RFID readers, beacons
- Click streams from a web server
- Prices – trades and quotes – from the financial markets
- Social media: tweets, posts, check-ins
- User actions in an application
- Transactions and workflow events
- IT systems logging events

In short, any device or application that produces information when things happen, or else to regularly report status, is a potential source for an event stream.

Immediate Insight

While access to all this information represents an opportunity, it also represents a challenge. Actually two challenges:

- How to cope with this ever-growing stream – make that flood – of information
- How to turn raw data into useful information – how to understand and act on the information before it’s too late

This is what's often referred to as the “Velocity” aspect of Big Data. New data is constantly arriving – at high speed and in high volume – and you want to find the information buried in the noise. There's value in understanding what's happening as it happens. But how?

The answer: smart data streaming.

- Receive and process event streams from one source or thousands of sources
- Process the data as fast as it arrives – even if that means hundreds of thousands or millions of events per second
- Filter out the noise, according to simple, complex, or even dynamic criteria
- Capture the useful data in the HANA database – in the optimal form
- Apply continuous queries to monitor trends and correlations, watch for patterns or compute summary information
- Generate alerts or initiate an immediate response
- Continuously stream select information to live dashboards
Complex Event Processing

Consider the following situations:

- Sensor data from equipment is monitored for trends or correlations that indicate a problem, alerting an operator to take immediate action before equipment damage occurs.
- A commodity pricing application continuously adjusts quoted prices in response to market conditions – where delays mean either lost business or lost profit.
- IT system events are continuously monitored to watch for patterns that indicate a possible security threat.
- User actions on a web site are analyzed to determine the best offers to show, not just based on historical data for the user but also considering current context.

These are just a few examples of the types of applications that can benefit from event stream processing, a type of complex event processing (CEP). The common denominator of these applications is that they share the need to continuously collect, process, and analyze data in real time, producing results without delay, even when the data arrives at very high rates. CEP technology analyzes incoming events in the context of other events as well as other available information, turning raw data into useful insight.

Examples of Solutions using Event Stream Processing

Organizations use event processing technology in a wide range of different applications across a number of industries. Here’s just a sample of some of the types of applications that are leveraging event stream processing for real-time insight and response.

Internet of Things (IoT)
- Actively monitor data arriving from sensors and smart devices
- Generate alerts when immediate attention is warranted
- Predictive maintenance: alert operations staff to imminent failure
- Situational marketing: target offers to users based on context
- Consolidate live status information in operational dashboards
- Collect high-value data in SAP HANA for use by application as well as historical analysis and reporting

Telecommunications
- Real-time system and network monitoring – Traffic volumes and problem detection
- Fraud detection – Spotting unusual traffic patterns in real time
- Customer retention – Proactive customer service
- Marketing – targeting offers to context

Manufacturing
- Predictive maintenance
- Process optimization
- Operational monitoring and alerting

Energy
- Short-term demand forecasting
- Smart-meter monitoring
- Demand management and response
- Oil and gas production monitoring

Retail - eCommerce
- click-stream analysis for real-time offer optimization
- fraud detection
- customer experience management

Financial Services
- Trading and risk: real-time valuation and exposure monitoring
- Market data aggregation
- Trade monitoring
- Real-time index calculation
- Fraud detection and prevention

Public Safety
- Crisis management – alerting and updating operational dashboards with data from the field
- Infrastructure monitoring – early warnings by analyzing sensor data in real-time
Basic Features of a Streaming Project

Regardless of the specific business application, all streaming projects are used to perform one or more of the following functions.

**Data Collection:** Capture event data in the SAP HANA database. Beyond simply capturing raw data, business logic can be applied to the incoming data to filter, transform or enrich the data, capturing useful data in the most meaningful form. ESP can also be used to help manage the size of the data set being captured. For example: high value data can be captured in SAP HANA, while the rest is diverted to Hadoop or simple event log files. ESP can also apply rules to correlate events, capturing compound rather than individual events, or can sample the data.

**Situation Detection:** Incoming events are monitored to detect trends or patterns that indicate the existence of an immediate opportunity or an imminent problem. This can range from a simple filter to a complex set of rules that aggregate or correlate incoming events. When the situation is detected, a high-level (complex) event is instantly published – with information about the situation. This can be sent as an alert to users or can be used to initiate an immediate response.

**Continuous Computation:** Incoming events are correlated, grouped, and aggregated - and computations are then applied to produce new information such as summary data, high-level statistics, or adjustments to key operating parameters. This data can be displayed on live operational dashboards, can be available to applications for immediate on-demand access, or can be sent to applications or systems to adjust current operations.

Smart Data Streaming: System Landscape

Smart data streaming runs on one or more dedicated hosts within the SAP HANA system landscape. Generally two or more streaming hosts are recommended for resiliency, though a single streaming host can be configured if desired.
The Streaming Cluster

Streaming projects run on the streaming server – or “cluster”. This can be a single machine or a multi-machine cluster. The streaming cluster runs as part of the HANA system and there can only be one streaming cluster for a single HANA system (SID).

The cluster is managed by the streaming cluster manager(s) that receive and manages requests to start and stop projects. Each project runs in its own container – an independent server process that is started by the cluster manager. The cluster manager is also responsible for restarting projects after a failure of the project or the machine it’s running on. Any number of cluster managers can run – all are peers, which avoid a single point of failure.

Streaming projects, running on the streaming cluster, connect to the SAP HANA database via ODBC.

Streaming Projects

Streaming “projects” are compiled into an executable that runs on the streaming cluster. The project defines the input streams and the set of continuous queries or other operations that will be applied to the incoming data to produce the desired output. Project may also include adapters to connect to data sources, connect specific output streams/windows to HANA tables, and to publish output to other destinations.

Streaming projects are defined in an event processing language called CCL (continuous computation language) which is derived from SQL. Don’t get the wrong impression from the use of SQL, however. Projects defined in CCL do not run on the database but rather are compiled into executables that run on the streaming cluster. Later in this document we’ll take a closer look at CCL.

So now let’s look at the building blocks and underlying concepts that make up a streaming project:

Smart data streaming is designed to process and analyze streams of events. In the real world, an event is something that happens. In an event-stream-processing context, an event is actually a message that contains information about a real-world event. An event could be an order arriving in a database, a price update on a stock market ticker, a user clicking on a web link, or a sensor reporting a piece of information.

Each event (message) contains one or more fields, with each field containing some information about the event. Applying database terminology, an event is a “tuple.” In fact, because CCL is derived from SQL, we refer to the fields in an event message as columns. And each field or column is assigned to a specific datatype.

Events flow from a source into a streaming project. The entry point in the streaming project is either an Input Stream or an Input Window. Each input stream/window has a defined schema – a set of named and typed columns – and can only process incoming events that have a compatible schema. Adapters can be used to connect the input stream/window to the source, or the source can initiate a connection.
Streams in a CCL project are stateless: that is, a single event enters, passes through the stream, and is “forgotten” once it has been processed. But you are limited in the types of analysis you can do on a single event. Therefore CCL lets you easily create event Windows.

Windows in a CCL have state. They either contain recent events or information that has been computed based on recent events. They are very similar to a table in a database, except that they are constantly changing. Incoming events add a row to the window, update an existing row in the window, or delete a row from the window. And when we say they contain recent events, or information about recent events, “recent” is relative. The size of the window can be defined based on time or size and could be very small – e.g. 10 events, a few seconds, or event milliseconds - or quite large – e.g. 10 million events, 5 days, etc. The key point is that windows are transient, holding recent information to support the desired computations. They are not designed for permanent persistence and a window should be attached to a HANA table when permanent persistence is desired.

The size of the standard CCL window can be set in three ways:

- **Size** - number of rows – A window set to a fixed number of rows will shed older rows when new rows are added once the size limit is reached.
- **Time** – A window with a time policy will keep rows for the defined time period and then delete them. So a one-minute window will keep each row for one minute after it arrives (or is last updated) and then delete it. (or, in the case of jumping windows, will collect events for a given time period, then delete all the events from the window and start collecting again)
- **Self-managed** – “KEEP ALL” – where a window is processing deletes in addition to inserts, a window can be self-managed. No size or time limit is set, but rather incoming events will delete existing records in the window. An example of this could be an order book that has a set of open orders. New orders are added to the book, and when an order is filled or canceled, an incoming “cancel” or “fill” event removes it from the book.

There is also the ability to create custom windows using other processing logic to manage what goes into the window and/or what gets removed from the window.

All windows have **primary keys**. A primary key consists of one or more fields in the row that uniquely identifies that row in the table.

Events can (optionally) have explicit operators, or “opCodes.” Incoming events may directly update information maintained in a window, and they can contain an opCode that indicates how they should be applied to the window. Thus, an event with an opCode of “insert” adds a row to the window, an “update” event updates an existing row with the same key, and a “delete” event deletes the row with the indicated key value.

Continuous queries and custom flex operators take input from one or more streams or windows and apply an operation to produce a new (derived) stream or window. Streams and windows that are produced as the result of a continuous query can be output streams/windows or they can be local streams/windows, performing an intermediate step on the overall processing of the event. A single project can have any number of output streams/windows and in fact all the streams and windows in a project can be made available for subscriptions.

Any output streams or windows can be published via adapters. They will write output events, in real-time, to the configured destination. Output streams and windows can be connected to HANA tables, such that the output is written to the HANA database, either inserting every event, or applying inserts, updates and deletes – depending on how it’s configured. In addition, client applications can initiate subscriptions to output streams and windows.

Streaming projects can connect to other streaming projects via inter-project bindings. This allows the output of a stream/window in one project to be connected as input to another project. The binding can be initiated by either side and can connect projects regardless of where they are running.
Design Time Tools

The streaming design time tools provided as a plug-in to the SAP HANA Studio. Once installed, the streaming plug-in adds two new perspectives: the Streaming Development perspective and the Streaming Run-Test perspective.

Note that the streaming plugin is not shipped with the standard SAP HANA Studio install or the SAP HANA studio plugin for Eclipse; the streaming plugin is downloaded and installed separately and can be installed at the same time you are installing the SAP HANA Studio or it can be added to an existing studio installation.

The streaming plugin for HANA Studio includes the following tools:

- Streaming project workspace – view, open and manage all the streaming projects you have created or are working on. New projects are automatically added and you can import existing projects
- Two streaming project editors: the visual editor and the CCL (text) editor. Project can be created, edited and viewed in either – and the user can switch between them in an open project
- The streaming server view that lets you add connections to any/all of your backend systems
- The ability to compile, load and run a project on the server and the ability to start/stop existing projects on the server (subject to suitable permission of course)
- A suite of test tools to test your streaming projects:
  - Stream Viewer: view contents/output from streams and windows
  - Event tracer and debugger (set breakpoints) to see how events are being processed in the project
  - Performance monitor: monitor throughput, queuing and cpu utilization across individual operators – identify bottlenecks
  - Record/playback: use the playback tool to stream in data from files, simulating a live source. Playback files can be created manually or can be created by recording live streams. Playback can use original speed (when playing back recorded data), play the data back as fast as possible, or use a fixed (adjustable) rate.
  - Manual input: manually generate an event and “publish” it to an input stream
  - SQL query tool: run SQL queries against streaming windows to check window contents
Interfaces and Adapters

One of the powerful aspects of smart data streaming is the decoupled nature of an event-driven architecture. Integration into an existing environment is typically non-intrusive. It can go in as an overlay to existing systems without needing to replace or reengineer those systems.

Connecting a streaming project to a data source or destination is typically done via input and output adapters. Smart data streaming comes with a set of common adapters that are part of the base package. It also includes publish/subscribe libraries for Java, C++ and .NET as well as an Adapter Toolkit.

Each adapter runs as separate process, and connects to an ESP project via a TCP socket. Adapters can run on the streaming cluster or they can run on remote hosts.

Standard Adapters

When using a pre-built adapter, it can simply be attached to a streaming project and configured to connect to the relevant source or destination. We are often adding new adapters, so check the developer center on SCN for the latest list. Standard adapters include:

- Message buses – Connect to a message bus to receive incoming events or publish events. Supported message buses include Java Message Service (JMS), TIBCO, and IBM Web-Sphere MQ.
- Databases – Load data from or publish data to databases via ODBC or JDBC. Most major databases are supported. Input data can be loaded at start-up, or the database can be periodically polled.
- Web Services: SOAP, REST and WebSockets
- File – Read input events from a file; write output events to a file, with CSV, JSON and XML parsing, one time or polled.
- Socket – Receive events on a socket or write events to a socket, with CSV, JSON and XML parsing.
- SAP RFC – call an SAP RFC to retrieve data on the input side; call an SAP RFC to deliver events on the output side
- Email (output only) – send an email each time an output event is emitted on a particular stream

Custom Adapters

A custom adapter can easily be developed using the Stream Adapter Toolkit. The toolkit provides a Java framework that supports pluggable transport and parse/format modules. The toolkit ships with a collection of pre-built modules and custom transports or parsers can be written in as little as a few hours.

Software Developers Kit

The Streaming SDK is available for Java, C, C++, and .NET. It supports both publish and subscribe capabilities to allow custom adapters to be built. What's more, while we generally talk about "adapters," in fact the pub/sub library can be embedded directly in an application, providing the application with native connectivity to streaming projects.

The SDK is designed to provide a simple way of:

- Connecting to an ESP cluster
- Constructing an event with a schema corresponding to the stream or window that will receive it
- Publishing an event to a specific stream or window in a specific project
- Subscribing to an output stream or window in a live project
- Parsing the event received from an output stream or window
**Project Deployment**

Streaming projects are normally stored in the SAP HANA repository, though project artifacts are simple text files that can also be stored directly on the file system or in an alternate version control system.

Streaming projects can be deployed on the streaming cluster in one of 3 ways:

- As part of a HANA Delivery Unit. Streaming projects in the HANA repository can be included in a delivery unit and will be deployed and activated when the DU is deployed.
- From the HANA Studio: simply compile the project, select run, and choose which cluster (if the Studio is connected to more than one) and which workspace to run it in.
- From the command line or from scripts using the command line utilities.

**CCL: Continuous Computation Language**

CCL is the event processing language of smart data streaming. Streaming projects are defined in CCL, which defines the inputs and a set of continuous queries to produce the desired outputs.

CCL is an adaptation of SQL with extensions for event streams. It provides the same type of sophisticated data analysis functionality you find in SQL, including the ability to filter, group, aggregate, and join data across streams. However, CCL also includes features that are required to manipulate data during real-time continuous processing, such as defining windows on data streams, watching for patterns of events, and invoking custom event handlers.

What really sets CCL apart from SQL is that CCL is designed for expressing continuous queries. A normal SQL query against a relational database executes once each time it is submitted to a database server and must be resubmitted every time a user or an application needs to re-execute 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 on the streaming server, a registered query executes each time data arrives from one of its data sources.

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

**Common CCL Query Types**

A CCL project consists of one or more input streams or windows and one or more derived streams or windows. Each derived stream or window includes one or more continuous queries that take data from inputs and produce the derived result.

Most continuous queries do some combination of the following operations:

- **Filter** – Filter an input stream, only passing through the events that pass the filter criteria.
- **Aggregate** – Group incoming events according to a common “key” value, producing a single output record for each group. One or more fields in the output record will typically be computed using aggregate functions such as `sum()`, `count()`, `average()`, or even more advanced statistical functions such as standard deviation or standard error.

### Input Stream:

```sql
CREATE INPUT STREAM EventsIn
SCHEMA (
    ID INTEGER,
    Value Integer,
    TS msdate);
```

### Filter:

```sql
CREATE OUTPUT STREAM Filter1
AS SELECT * FROM EventsIn WHERE EventsIn.Value > 28;
```

### Aggregate:

```sql
CREATE OUTPUT WINDOW MovAvg
PRIMARY KEY DEDUCED
AS SELECT EventsIn.ID ID,
        avg (EventsIn.Value) Value,
        EventsIn.TS TS
FROM EventsIn KEEP 3 MIN
GROUP BY EventsIn.ID;
```

### Join:

```sql
CREATE OUTPUT STREAM Join1
AS SELECT
    Dev2.ID ID,
    Dev2.Temp Temp,
    MovAvg.Value Value,
    MovAvg.TS TS
FROM Dev2 INNER JOIN MovAvg
ON Dev2.ID = MovAvg.ID;
```
correlation.

- **Compute** – Transform an incoming event into an event with a new set of fields where the new fields are computing from the fields of the incoming event, possibly along with other information.

- **Join** – Join events or records from two or more inputs based on common field values. You can join a stream to a window, join two windows, and even do multi-way joins. This can be used to enrich incoming events with additional reference information, or it can combine data from multiple event streams, matching the events within a defined time window.

- **Pattern Detection** – Watch for a specific pattern of events, and generate a new event when the pattern is detected. Patterns are defined with a time interval and a sequence or combination of events, and they can include missing events – for example, a pattern might be “eventA followed by eventB, but NOT eventC – within 10 minutes.”

**The CCL Query graph – Data-flow Programming**

A streaming project can consist of a single input stream followed by a single continuous query, but a project will often contain multiple continuous queries. Large projects may contain dozens of inputs and hundreds of continuous queries.

Smart data streaming uses data-flow programming to direct incoming data into and through a set of continuous queries. Thus, a project is broken down into a sequence of continuous queries, each of which further refines the incoming data into the set of desired outputs. It’s very easy to represent this visually (using, for example, the visual editor) as a set of streams and windows, with the data flowing between them. Thus, a typical query graph of a streaming project might look like Figure 5.

**CCL Script for Versatility**

SQL is a great starting point for the most common functions used in processing event streams. It has the benefits of familiarity and the simplicity of a declarative language. But there are times when the event processing logic you need to implement can’t be easily expressed in an SQL query. That’s where CCL Script comes in. It brings extensibility to CCL.

CCL Script is a simple procedural scripting language that has a syntax similar to Java, but its spirit is closer to languages such as AWK or Perl that solve relatively small programming problems. It provides procedural
control, the ability to iterate using WHILE and FOR loops, and data structures for holding data from one event to the next.

**CCL Script Controls and Data Structures**

CCL Script includes controls such as FOR, WHILE, and IF, giving the user complete procedural control with the ability to iterate over data sets. CCL Script includes the ability to define variables that retain state from one event to the next. It also includes advanced data structures that are designed for efficient handling of events and event sets, such as:

- **Event Cache** – An alternate windowing mechanism for working with sets of events that allows greater control than the standard CCL KEEP clause
- **Dictionaries** – Sets of key or value pairs for efficient lookup
- **Vectors** – Ordered sets of values or data structures

**Flex operators**

Flex operators are custom, programmable event handlers that go beyond what you can do with the standard relational operators of CCL. They are used to create a new derived stream or window in the same way that a continuous query does. However, instead of passing the incoming event through a SELECT statement to produce the result, a flex operator invokes an event handler written in CCL Script upon the arrival of a new event. Thus, a user can write sophisticated custom operators in CCL Script and not limit processing to the set of operations that can be achieved via the standard SQL relational operators.

**Built-In function library**

Each continuous query will include a set of “column expressions” to compute the value of each field in the new event being produced. These expressions can draw from the extensive set of built-in functions:

- Math
- Statistical analysis
- Geometry
- Logical
- String
- Date and time
- Calendar
- Aggregation
- Sets
- Bitwise
- XML

**Modularity**

A streaming project can invoke external CCL modules. This facilitates reuse and team development, where different team members can be responsible for different modules. It can also be used to improve the maintainability of large complex projects.

A project can use a CCL module defined in another CCL file by first using the IMPORT command to include the CREATE MODULE statement from the external file. It then uses the LOAD MODULE statement to invoke the module, and a single module can be invoked more than once within a project. When a module is loaded, the input and output bindings are set. Additionally, any parameters used by the module are set when the module is loaded. Thus, a module that computes a moving average could have the duration set via parameter such that it could be loaded one place to produce a 10-minute moving average and loaded another place to produce a two-hour moving average.
System Administration

The smart data streaming cluster can be monitored and managed via the standard SAP HANA administration tools.

- Each streaming cluster node will be listed in the Administration Console in the SAP HANA Studio, along with status information
- Streaming cluster tiles can be added to the SAP HANA Cockpit. Drill down provides status information on individual projects and adapter

Security

Smart data streaming has built-in security features for authentication, access control, and encryption. User authentication is typically done by the HANA system, though Kerberos authentication is also supported. Secure socket layer (SSL) encryption is available to protect communication between streaming projects client applications (including both data sources and consumers).

Role-based access control can restrict what individual users or client applications can do. Authorization can be used to restrict activities at both the cluster level and the project level; within a project, it can be used to restrict access to individual streams and windows. Thus, based on the users’ authenticated identity, access control can restrict whether they can:

- Administer the cluster
- Start or stop projects
- Write to or subscribe to a particular stream or window in a named project

Resiliency

At the level of the streaming cluster, as long as there is more than one machine in a cluster, there is no single point of failure. While a cluster can run as a single node on a single machine, where resiliency is desired there should be at least two machines in the cluster, each with a cluster manager.

When starting a project on the cluster, the project is normally started with Automatic Failover enabled. In this mode, if the project fails, the cluster manager will restart it. Projects can also be started with auto recovery disabled, in which case a project that stops for any reason will need to be manually restarted.

Guaranteed Data Delivery

Some use cases need to ensure that no data is lost, even in the event of a failure. Smart data streaming has features that can be used to ensure end-to-end delivery of every message. These features include:

- Recoverable windows: windows in streaming projects can be backed by a disk-based log store that allows the window to be re-built after a project restart. All the windows in a project can be recoverable, or just the ones that matter.
- When a window is rebuilt, any downstream data structures will be rebuilt
- Publisher commit: publishers can explicitly issue a commit statement at regular intervals. When the publisher receives confirmation that the commit completed, all events written to recoverable windows are confirmed.
- Publishers must be prepared to re-send any events should a project fail before the commit was completed
- Commits can be issued after every event, but there will be a performance impact
Subscriptions to output from streaming options can use the “Guaranteed Delivery” feature. Output events will have a sequence number and the subscriber will periodically confirm receipt of all events up to the current sequence number. Should the subscriber fail or lose the connection to the server, upon reconnect any events that were published after the last acknowledgement point will be resent. (Note that this provides “at least once” semantics. Subscribers must be able to receive duplicates)

An error stream can be used to capture any events that cannot be processed due to run-time errors (e.g. an unexpected value in a field that prevents the event from being processed). The error stream can be logged as well as supporting real-time subscriptions.

Scalability and Performance

Smart data streaming was designed from the ground up for speed and scalability - capable of meeting the performance needs of the most demanding applications. A streaming project on a relatively small host can easily process over 100,000 messages per second. With adequate compute resources, streaming projects can scale to process well over 1 million events per second on a single server.

Latency, measured from the time a message arrives until processing is complete and results have been published, is typically on the order of a few milliseconds – as measured end to end, including I/O.

To maximize throughput and minimize latency, smart data streaming uses a real-time data-flow architecture that moves the data through a directed graph of continuous queries. Unlike a database query that first requires the data to be stored before the query can be run, the entire streaming architecture is event-driven and designed to flow data through the system as quickly as possible. All data is held in memory, and even where windows need to be recoverable, a high-speed disk-based log store is used to provide data recoverability with minimal impact on performance.

The streaming server has also been designed to deliver consistent latency. When latency is measured, some systems may display low average numbers but with a wide range of values, including spikes that can run into seconds. For applications where latency is critical, smart data streaming can be counted on to deliver a consistent latency profile.

There are a number of architectural features that come together to deliver this scalability and speed:

- **Multi-threading**: allows for both pipelining and parallel processing. Each streaming project is multi-threaded, allowing a single project to process multiple events simultaneously
- **Partitioning**: CCL includes a partitioning feature that can create multiple parallel instances of an operator or module, with automatic partitioning of the stream(s) flowing in using a choice of hash partitioning, round-robin, or a custom partitioning expression
- **Direct connections**: while streaming projects run on a streaming cluster, each project runs as a separate process and publisher and subscriber connections are direct to the project via TCP sockets, with data transferred in an efficient wire format
- **Inter-project bindings**: large projects can be split into multiple smaller projects that are interconnected via project bindings. Bindings can connect projects running on the same host or on different hosts.
Unique Features

There are a number of features of the SAP HANA smart data streaming architecture that are typically not found in event processors:

- **HANA tables in streaming projects** – join event streams to HANA tables and views. The CCL Reference element serves as a proxy to the HANA table/view, pulling in data as needed, and optionally caching the data. Includes cache properties to control age/refresh.

- **Native state management** – Incoming messages can be processed as inserts, updates, deletes, or “upserts.” This lets a streaming project very efficiently manage data windows where incoming messages don’t just represent a new data point in a time series but instead represent an update to previous information. Most CEP implementations don’t handle updates and deletes – they treat all incoming messages as new data points in a time series. The reality is that many data streams produce updates, changes, and cancellations. Whether it’s a change to an order book or a correction to data previously sent, these updates need to be applied to previously received data to maintain an accurate view of the current state.

- **CCL Script** for versatility, while retaining simplicity. CCL, derived from SQL, provides familiarity and ease of use. But declarative SQL can be limiting. CCL Script provides extensibility, making it easy to add powerful custom operators and functions with the fine-grained control of a procedural scripting language and with data structures for efficiently maintaining state between events.

- **On-demand queries** – ODBC driver to execute snapshot SQL queries against windows in streaming projects. Includes support for SAP HANA smart data access.

- **Rich subscriptions** – Each subscription to an ESP output stream can be tailored to the needs of the consuming application. Subscriptions to windows can deliver the full window followed by updates, or just updates. Subscriptions can include an SQL SELECT statement to further filter, aggregate, or otherwise refine the data.

- **Security** – Built-in security including access control, authentication, and encryption

Related Content

- [SAP HANA smart data streaming - product documentation](#)
- [SAP HANA Academy: smart data streaming playlist](#)
- [Introduction to SAP HANA smart data streaming](#) – links to get you started
Copyright

© 2014 SAP SE SE or an SAP SE affiliate company. All rights reserved.
No part of this publication may be reproduced or transmitted in any
form or for any purpose without the express permission of SAP SE.
The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components
of other software vendors. National product specifications may vary.

These materials are provided by SAP SE and its affiliated companies (“SAP SE Group”) for informational
purposes only, without representation or warranty of any kind, and SAP SE Group shall not be liable for
errors or omissions with respect to the materials. The only warranties for SAP SE Group products and
services are those that are set forth in the express warranty statements accompanying such products and
services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP SE and other SAP SE products and services mentioned herein as well as their respective logos are
trademarks or registered trademarks of SAP SE in Germany and other countries.

Please see
for additional trademark information and notices.