How Streaming Analytics Enables Real-Time Decisions
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>What Is Streaming Analytics?</td>
<td>1</td>
</tr>
<tr>
<td>How Does SAS® Event Stream Processing Work?</td>
<td>2</td>
</tr>
<tr>
<td>Overview</td>
<td>2</td>
</tr>
<tr>
<td>Event Stream Application Development</td>
<td>3</td>
</tr>
<tr>
<td>SAS® Event Stream Processing Studio</td>
<td>4</td>
</tr>
<tr>
<td>How Does Event Stream Processing Use SAS Analytics?</td>
<td>5</td>
</tr>
<tr>
<td>Enriching Events</td>
<td>5</td>
</tr>
<tr>
<td>Applying Advanced Analytics</td>
<td>5</td>
</tr>
<tr>
<td>Conclusion</td>
<td>6</td>
</tr>
</tbody>
</table>
Introduction

The idea of processing data from events as they are occurring is not new. Many platforms already have ways of generating alerts when certain thresholds are reached or when specific data is available.

Systems can track message queues or use web services, for example, and take action when needed, but often these systems are reactive in nature. They are focused on the consumption of pre-existing information, listening for a response or awaiting instruction. These events occur as the result of an action or set of actions – a purchase, a payment, a failure or some other trigger.

Technology has advanced so that event data is constantly generated with more precise detail. An event can now be something that happens within a system at a point in time, such as a click, a sensor reading, a tweet or some other incremental, measurable activity. Message queues and similar architectures have difficulty efficiently managing and tracking consumption of these event streams, not to mention handling their velocity and the volume of data.

Another distinguishing characteristic of event stream data is time. Time is crucial to define context and is the common factor across all types and formats of streaming data.

A single event is typically not informative. But when combined with other events that occur in proximity to it, a story emerges. These events could be a potential machine failure, a case of fraud, a potential cyberattack, an opportunity to drive an existing sale, or a chance to provide a better customer experience.

The type of real-time analytics required in these instances needs a specific infrastructure, one that can continuously analyze an ever-changing set of data-driven events. The architecture must be able to capture events, assess them, make decisions and share the outputs all within the confines of specific time windows.

Event stream processing delivers the architecture needed to analyze streaming data, providing immediate situational awareness. It enables you to proactively respond to changing conditions to improve operations and foster more successful customer interactions.

What Is Streaming Analytics?

Streaming analytics is the application of analytics to data while it’s in motion, and before it’s stored – and includes data manipulation, normalization, cleansing and pattern of interest detection. Streaming analytics affords insights into:

- Social networking activities.
- Data streams from satellites.
- Devices.
- Networked machines.
- Sensors.
- Internet systems.
- Connected things (IoT).

With streaming analytics, the goal is to analyze data as close to the event location as possible, before its value is diminished due to information lag and before the volume of data overwhelms traditional analytics. This approach enables you to identify and examine patterns as events occur, so you can take immediate action on those events of interest as they happen.

Managing data in motion is different from managing static data (also known as data at rest). To deal with data in motion, event stream processing relies on:

- **Assessment**
  With massive volumes of streaming data, it is simply not practical to store it all. Much of what is generated is irrelevant for any action, analysis or even archiving. Event stream processing can be used to standardize the incoming data, determine if the data is relevant and if any downstream processing is needed. If not, the event can be discarded without taking up processing bandwidth.

- **Aggregation**
  Event stream processing is used to continuously calculate metrics across defined intervals of time to understand real-time trends. This type of continuous aggregation would be difficult with traditional tools because of the time lag and I/O overhead associated with storing data on disk and then running a query.
• **Correlation**
  An individual event may not be significant as a single data point. But when you can establish its relationship to multiple events from a single stream or even multiple streams, then you can set the context to assess these events. Monitoring patterns and correlations (such as identifying that Event A was followed by B and then C within a given time window) provides more accurate understanding of a situation than analyzing Event B in isolation.

• **Analysis**
  One of the advantages of streaming analytics is gaining as much insight as possible while the data is still in motion. Scoring or performing advanced analytics while events are happening ensures that operational processes are tied with existing conditions. Adding out-of-stream (i.e., traditional) analytics to events from data streams enables you to further enrich events and put them in a business context. This approach can help you to use event stream processing to create additional revenue, improve a customer experience or reduce costs and risk.

How Does SAS® Event Stream Processing Work?

**Overview**

SAS Event Stream Processing, our streaming analytics solution, includes a server, which provides the run-time environment for executing the event stream models against one or more data streams.

The run-time environment is built on C++. An interactive GUI or an optional XML interface are available for model development and manageability. SAS provides the most complete and integrated solution available using a publisher/subscriber interface to connect to streaming sources and output locations.

There are three main elements to the architecture (see Figure 1):

• A **publishing interface** for connecting to live data streams. Connectors publish the event streams into source windows. Published operations natively read event data from a specified source and place that event data in a source window of a running event stream processor.

• The **SAS Event Stream Processing server** applies an event stream model that specifies how streaming data is meaningful to output location subscribers. The server supports continuous queries that are defined in the model. A continuous query can be thought of as a directed graph, which is a set of connected nodes (windows) that travel down one or more parallel paths. These nodes represent the pattern definition, transformations and analytics that are applied to the data (think of a flow diagram).

  The server supports multiple languages to define expressions, patterns and transformations. You can use embedded aggregation functions, or you can define your own procedural code (using C++, SAS DS2 or the DataFlux® Expression Language). In addition, you can define windows that capture streaming data over a period of time – static or rolling – and check for patterns or score advanced analytic models and/or text analytic models based on the aggregate events that have been cached in the time window.
A subscriber interface shares the results with other systems or information consumers.

Adapters and connectors can be subscribed to event stream windows. The subscribed operations natively write output events from a running event stream processor window to the specified target.

Event stream processing engines can be embedded in new or existing applications. SAS Event Stream Processing enables continuous analysis of flowing data when low latency and incremental results are important.

SAS Event Stream Processing applications can analyze millions of events per second, with latencies in the milliseconds - even on small, commodity servers with 4-8 cores. Because of the lightweight architecture and thread pool management, it is easy to linearly scale the environment to run multiple SAS Event Stream Processing servers in a grid to distribute the load and extend in-memory resources.

Our solution also offers high availability and provides this capability through a software-based approach for native failover. Native failover ensures that data streams are never lost because servers can automatically switch over and reroute traffic to active servers when failures are detected. SAS Event Stream Processing provides 1+N way failover using message bus technology provided by Solace Systems, Tervela or RabbitMQ.

Event Stream Application Development

Conceptually, an event is something that happens at an identifiable time that can be recorded (as a collection of fields that have a definable schema).

In designing an application, programmers need to know:

- What streams of data define an event (i.e., how do the event streams need to be published)?
- What happens to that data? How are event streams transformed and modeled?
- What is the output of the data streams? What analyses are consumed by which subscribers and in what format do they need them?

The answers to these questions determine the structure of the event stream processing model. There are several parts to consider (see Figure 2).

Each event stream application uses a single model. Within that model definition, you can have one or more projects. Projects define the dedicated thread pools (collections of threads on which work can be scheduled).

Using a thread pool in a project enables the event stream processing engine to use multiple in-memory processor cores for more efficient parallel processing. This gives the modeler the ability to optimize the performance of the environment across multiple projects.

---

**Figure 2. SAS Event Stream Processing application architecture.**
Within each project, you can also have one or more continuous queries. Each continuous query consists of at least one source window and one or more derived windows.

The source window defines the schema for the input stream. The derived windows determine the actions to take from the analysis. Derived windows can detect patterns in the data, transform the data or perform computations based on the data, and, this is important, they can be connected to other derived windows.

Here are some examples of derived windows that you can use:

- **Aggregate**: computes aggregations of non-key fields.
- **Compute**: performs one-to-one transformation of input events to output events.
- **Copy**: defines the retention policy for data based on the parent window. The parent window can be the source window or a derived window. Retention can be defined in static or rolling time periods or by number of events.
- **Counter**: counts the number of events streaming through your model and the rate at which they are being processed.
- **Filter**: defines which data is relevant to continue processing in-stream based on a Boolean filter function or expression.
- **Functional**: enables you to use different functions to manipulate or transform event data. When an event enters a functional window, the window looks for a function with a name that corresponds to each field in its schema.
- **Join**: joins two input windows and supports inner and outer joins.
- **Notification**: sends notifications through email, text message or multimedia message.
- **Pattern**: defines connections between different patterns of interest, based on both time and calculation.
- **Procedural**: enables you to define custom functions (including advanced analytics) to apply to input data.
- **Text**: enables the classification and extraction of terms and sentiment identification from an unstructured string field.

**SAS® Event Stream Processing Studio**

To simplify the development and implementation of event stream processing models, we created a web-based, point-and-click interface that supports model development and testing. The SAS Event Stream Processing studio enables you to visualize event stream processing models in a process flow, as you add windows to your model (see Figure 3).

From the visual interface, you have access to the derived windows. As windows are added and joined to each other in the model flow, the appropriate XML code is automatically generated. You can publish the resulting XML code to the server for execution, and you can import existing SAS Event Stream Processing models in XML to create the process flow diagram.

![Figure 3. Screenshot of SAS Event Stream Processing studio.](image-url)
How Does Event Stream Processing Use SAS® Analytics?

One of the important advantages of SAS Event Stream Processing is its use of SAS analytics. The value of using streaming data increases significantly when you can interrogate the events in a business context. There are two key methods:

- Enriching events with additional intelligence gathered by your organization.
- Applying advanced analytics, such as scoring models or other business rules, to the streaming data.

The streaming model results can be fed directly into an operations data store to examine events outside of expected conditions or archived for audit purposes. If deeper analysis reveals new patterns of interest, they can serve as new elements within SAS Event Stream Processing detection (see Figure 4).

Enriching Events

While events can provide insight over a specified time, there might be instances when the event is tied to a longer term process or relationship that continues beyond a live event window.

For example, if an event is triggered by a specific customer activity, such as when a customer enters a retail store’s Wi-Fi network, it’s possible that this customer has some history with the store. As a result of that history, there might be additional information about that customer, which could include historical buying behavior, product preferences or other segmentation characteristics.

You can capture this intelligence in the streaming model by joining events with existing data to enrich the live event data. Using connectors, you link data that is stored in a database, XML, CSV, JSON, or other sources, and load that into a SAS Event Stream Processing project. That historical data is combined with streaming events to provide richer context, in-memory and in real time.

Applying Advanced Analytics

SAS Event Stream Processing provides the ability to execute SAS analytics in the stream using the procedural window. The procedural window enables you to specify one or more input data streams and their execution code. This code is referred to as an input handler and can be written in C++, SAS DS2 or XML. The following code is an example of how to define a DS2 input handler that calculates the total cost of a stock transaction by multiplying stock price by number of shares (size):

```plaintext
input schema:
"ID*:int32,symbol:string,size:int32,price:double"

output (procedural schema):
"ID*:int32,symbol:string,size:int32,price:double,
cost:double"

ds2_options cdump;
data esp.out;
dcl double cost;
method run();
  set esp.in;
  cost = price * size; /* compute the total cost */
end;
enddata;
```

Figure 4. Injecting data enrichment, analytic models and business rules.
Although the example above is a simple calculation, it is possible to use any SAS DS2 code within the event stream. SAS DS2 code can be associated with predictive models, forecasts, data mining or other advanced algorithms.

Note that these advanced analytic models are developed using a larger set of data that includes history and is tested using a validation sample; thus traditional algorithm development is needed. These advanced analytics are developed outside of event streams, based on data at rest using solutions such as SAS® Enterprise Miner™, and easily deployed within the event stream model.

Traditional, in-depth analysis can identify new patterns or measures that need to be monitored and encoded into the streaming analytics for detection. This requires a feedback loop from out-of-stream (at rest) analytics to in-stream (in motion) analytic scoring so that models are enhanced over time as events evolve.

**Conclusion**

Organizations are beginning to take a deeper look at capturing data about streaming events. Being able to act as soon as events are generated improves operational responsiveness and organizational effectiveness. The challenge is moving beyond the limited value that static data assessment provides.

As context is applied to events in real time, the opportunity to act in a relevant and precise manner is vastly improved (see Figure 5). Organizations will have a competitive advantage if they use the power of event streams to immediately take the best action.

SAS Event Stream Processing provides the flexibility and performance to apply rich context to the events as they occur. It provides the full range of contextual intelligence to immediately act with confidence by supporting:

- Integration with SAS® Analytics.
- A visual interface to streamline development and implementation of models.
- Extremely high throughput and resiliency.
- Highly flexible architecture to integrate with streaming data sources and output channels.

To learn more about streaming analytics, download the white paper: **Understanding Data Streams in IoT**.
