Dipping your toes in the event stream processing pool

By: Mark Thomas, SAS Global Professional Services
Release Information

Content Version: 1.0 March 2015.

Trademarks and Patents


SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

Statement of Usage

This document is provided for informational purposes. This document may contain approaches, techniques and other information proprietary to SAS.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>SAS Event Stream Processing Engine Model Basics</td>
<td>1</td>
</tr>
<tr>
<td>VWAP Sample XML Model</td>
<td>2</td>
</tr>
<tr>
<td>Installation</td>
<td>3</td>
</tr>
<tr>
<td>Execute a Model</td>
<td>3</td>
</tr>
<tr>
<td>Stream viewer</td>
<td>4</td>
</tr>
<tr>
<td>Final Thoughts</td>
<td>6</td>
</tr>
<tr>
<td>References</td>
<td>7</td>
</tr>
<tr>
<td>Resources</td>
<td>7</td>
</tr>
</tbody>
</table>
Introduction

SAS Event Stream Processing moves analytics and data transformation much closer to event origination than has been possible in the past. Whether the data is generated from sensors on oil drilling equipment, streaming transactions, the web, or from sensors on a car, SAS Event Stream Processing can be configured to be the next phase of processing that occurs immediately after an event is created or issued by a source.

The prevalence of data generated by the Internet of Things is here and is expected to grow dramatically over the coming years. Feeding this data directly to Event Stream Processing minimizes lag time and provides the ability to react with minimal delay. Studies have shown that the potential business value of data generated by sensors steadily diminishes over time. Imagine being able to predict that a multi-million dollar part will fail by analyzing sensor data with minimal lag time and scheduling a replacement before failure, thus minimizing outages, down time and possible post-failure repairs.

To better understand the value of minimizing lag time and reacting quickly to large volumes of data please read Steve Foerster's blog as well as the recent paper by Frédéric Combaneyerre cited in the Reference section of this paper.

SAS Event Stream Processing is currently delivering value to existing organizations and is often included with one or more SAS solutions to add the insight from data streams to industry applications. SAS Event Stream Processing has been used in conjunction with SAS Solutions to add streaming data to existing applications, providing front-end processing of streaming data. For example, Event Stream Processing, licensed separately, can be used with SAS High-Performance Risk to identify and track illegal trading patterns as they occur. Alternatively, it's been used as a front-end for a Visual Analytics deployment to load data into LASR for real-time visualization and advanced analytics.

Given that SAS Event Stream Processing is expected to play a prominent and active role in SAS'S's future, this paper describes some of the basics of model components and the steps involved in running an Event Stream Processing XML model. The XML Modeling Layer is the primary method of feeding event stream processing engine definitions, also known as models, into an event stream processing XML server. With the latest release of SAS Event Stream Processing an interactive visual interface is used to develop the model, with the XML generated automatically behind the scenes. And although it is possible to embed SAS Event Stream Processing into a C++ application, most application developers will likely take advantage of the XML Modeling Layer and XML server.

A SAS Event Stream Processing deployment supplies numerous sample models in the src directory within Event Stream Processing's home directory. These samples are a convenient way to become familiar with designing and executing Event Stream Processing models outside of the visual interface. The models in this directory cover both C++ and XML models. Each sample model typically contains sample data as well as documentation on purpose and usage. This article focuses on the VWAP (Volume Weighted Average Price) XML model found in the vwap_xml directory.

SAS Event Stream Processing Engine Model Basics

Analytics, data transformations and pattern recognition of event streams are some of the capabilities that Event Stream Processing (ESP) models provide. The latter part of Steve's blog on Event Stream Processing identified some of the key components of an XML model. Several of the key components are reiterated here.

The following is the official SAS definition of an Event Stream Processing model from the SAS Event Stream Processing User's Guide:

A model is a user specification of how input event streams from publishers are transformed into meaningful output event streams consumed by subscribers.

Events are streamed into a model from a variety of data sources using publishers, known as either connectors or adapters. Adapters are networked stand-alone executables that use sockets, and thus can reside on the same host or on a different host, whereas connectors are in-process classes built into the model. Both adapters and connectors use the publish/subscribe API to stream data into or out of a model. In addition to the standard connectors and adapters included with the product, it is also possible to publish/subscribe via Java or C using the API.

The core of a model is an engine. An engine may consists of one or more uniquely named projects. Dedicated thread
pools are managed at the project level allowing the application developer to specify the number of cores to be used.

Projects consist of one or more continuous queries. Continuous queries are conceptualized as directed graphs which define how data flows through the model. Data flows through one or more parallel paths of a continuous query that performs transformations on the data. A continuous query will contain at least one source window that feeds at least one derived window.

Event stream adapters and connectors are used to subscribe to windows in the model and publishers output the modelled data to a variety of standard data sources.

**VWAP Sample XML Model**

As noted earlier this example explores a sample model provided with the software. If we diagram the components of the VWAP XML sample model as defined above, the model hierarchy would appear as follows:

![VWAP Model from ESP Samples](image)

Modeling hierarchy of the VWAP sample model, shipped with SAS Event Stream Processing

From the diagram above we see that the model contains one publish connector, one source window, three copy windows, three aggregate windows, one compute window and four subscriber connectors (Publish/Subscribe APIs are not shown). In essence this model employs the CSV connector to publish the source file (event stream), it then creates three copies using the copy window, each with varying levels of data retention, followed by an aggregation for each copy window. In addition, it computes a throughput rate in the comp_win window. A connector then subscribes to each of the aggregation and compute windows and writes the results to a file. The list of adapters can be found in the SAS Event Stream Processing fact sheet, posted on the SAS website.
The VWAP model, like the other sample models, doesn’t stream live data but streams data using file system connectors to publish a Comma Separated Variable (CSV) source file to the source window. CSV connectors are also used to subscribe from windows and write to files. The required source file and output files that are referenced within the model are located in the $DFESP_HOME/src/vwap_xml directory, where $DFESP_HOME is the home directory for Event Stream Processing. For details of configuring connectors and adapters in order to connect to data sources see chapters “Using Connectors” and “Using Adapters” of the SAS Event Stream Processing User’s Guide.

Installation

Before we discuss executing a model, let’s first take a quick look at how to install SAS Event Stream Processing. In short, deployment is quite simple. It’s not built on Base SAS and thus doesn’t require the same prerequisites as other SAS models. All required packages, libraries and executables are included within the SAS Event Stream Processing product.

Once the software order is downloaded to a depot, SAS Event Stream Processing installation is completed from the SAS Deployment Wizard. Installation doesn’t require a plan file and only takes a few minutes. Of course a plan file will be required if SAS Event Stream Processing is being integrated with another SAS solution. The only steps required after installation is to configure environment variables and copy the license file to the appropriate location. There are two variables required for operation on each platform where SAS Event Stream Processing is installed. For example, $DFESP_HOME and %DFESP_HOME% identify the home directory for Event Stream Processing on Linux and Windows, respectively. The other environment variable adds the path to libraries and binaries. Finally, a valid license file must be copied to the etc/license directory. Once the environment variables are set and the license file copied the product is ready for use.

Execute a Model

Executing the sample VWAP model on the XML server is also straightforward. Simply logon as the account used to install SAS Event Stream Processing and issue the following commands1:

```
$sasinst@asserver01 ~]$ cd $DFESP_HOME/src/vwap_xml
$sasinst@asserver01 vwap_xml]$ ls

-sasinst@asserver01 vwap_xml]$ ls
total 93884
-rwxr-xr-x. 1 sasinst sas 11788 Nov 11 11:43 model.xml
-rwxr-xr-x. 1 sasinst sas 9330550 Nov 11 11:03 trades1m.csv

-sasinst@asserver01 vwap_xml]$ $DFESP_HOME/bin/dfespm_xml_server -model-file://$DFESP_HOME/src/vwap_xml/model.xml -http-pubsub 5557
```

The first command changes directory to the location of the model and source file. The second command shows the contents of the directory. The third command starts the XML server and loads the model and it executes immediately.

XML server output for the model should appear as below. At this point the model is executing but other than these messages sent to the SSH session, there is no further indication of progress. Notice that the model messages indicate the ports involved and the retention values for the copy windows.

1 This exercise was performed on a Linux server, but similar steps would be taken on Windows or Solaris
Since the model writes four files to the vwap_xml directory, it's possible to check progress by verifying that the expected files have been created. The XML server doesn't terminate automatically, even after it has processed all of the data. Of course this is by design as the continuous query in the model is waiting for additional data. The concept of event stream processing is that a model running on an engine is continually available for new data until specific action is taken to terminate a server. Therefore it is necessary to start a new SSH session to check on the files. Once the session is started, change directory to the sample model and examine the contents. Notice the four CSV files created on March 18 in the screen shot below. This verifies that event data flowed from the CSV publisher connector through the model and was written via CSV subscribe connectors to external files.

**Stream viewer**

In reality not all models will write to files because different output are desired, like alerts, system triggers or situational monitoring and other reports. One method of “tracking activity” is to use the stream viewer visualization feature. Stream viewing is a capability included with the installation package and enables you to subscribe to a running Event Stream Processing model and display the events streaming through it. Events are displayed in a table format with each row equating to an event (it is also possible to graph live events). The engine must be started with the `http-pubsub` parameter and port number either on the command line or within the XML file. Then in order to run stream viewer, you take action on one of the following:

- Copy $DFESP_HOME/share/tools/streamviewer to a local machine and open `html` with a browser
- Copy $DFESP_HOME/share/tools/streamviewer to a local machine and open `streamviewer.html` with a browser

Once the stream viewer is running, it's necessary to enter a URL that points to the Event Stream Processing publish/subscribe HTTP provider. Using our sample VWAP model, if we started our XML Server with `-http-pubsub 5557` then we would specify http://sasserver01:5557 in the window for the Event Stream Processing Streamviewer Server.

Once the server URL is set you can subscribe to the windows in the model. Available windows of the model are displayed
by selecting the “Show Model” icon:

Selecting the “Show Model” icon presents the following screen. This screen displays the project names, continuous queries and windows associated with each query within a model.

Finally, select each window to which you want to subscribe. The results should resemble the following screen shot. Each window within the browser will display events from their respective model window. If the data is visible in the subscribed windows then events are successfully flowing through the model.
At this point you have executed and validated a model.

Final Thoughts

In this paper I've introduced the basic components of an Event Stream Processing model to help you get started. With a quick review of Event Stream Processing installation, a simple example of XML model execution and defined the steps used to validate a model using stream viewer visualization. These topics begin to scratch the surface of SAS Event Stream Processing.

It's now possible to filter, aggregate and analyze large volumes of continuous event data with minimal lag. Moving analysis closer to sensor and event data with SAS Event Stream Processing allows businesses to react immediately, precisely and proactively - before big data is passed on to other analytic applications or storage repositories for more in-depth analysis.

And although substantial value can be attained with adding Event Stream Processing to ingest high throughput, high volume data into operational decision environments – deciding what to action, what to ignore and what to store, even greater business value can be achieved when it's integrated with a SAS solution, such as High Performance Risk – bringing real-time intelligence to strategic applications. SAS Event Stream Processing opens the window on what is happening now, SAS solutions provide insight into what has happened in the past and how to anticipate what may occur in the future. Models developed in SAS Solutions can be encoded into SAS Event Stream Processing as DS2 models,
keeping an analytical pulse on real-time events. And when non-conforming patterns of interest are detected, they can be examined with the benefit of historical data – to define new, emerging trends and new patterns to monitor in event streams.

References

Foerster, Stephen “You React So Quickly Do You Have ESP?” Available at: http://blogs.sas.com/content/sgf/2015/04/30/you-react-so-quickly-do-you-have-esp/


Resources

   (please call Technical Support to acquire key to unlock this documentation: North America: Call 919-677-8008 & outside North America: Contact your local SAS Office as per: http://www.sas.com/offices/intro.html )


More Information regarding SAS® Event Stream Processing can be found at:

http://support.sas.com/software/products/esp/index.html

And
