Paper SAS686-2017

Fighting Crime in Real Time with SAS® Visual Scenario Designer

John Shipway, SAS Institute Inc., Cary, NC

ABSTRACT

Credit card fraud. Loan fraud. Online banking fraud. Money laundering. Terrorism financing. Identity theft. The strains that modern criminals are placing on financial and government institutions demand new approaches to detecting and fighting crime. Traditional methods of analyzing large data sets on a periodic, batch basis are no longer sufficient. SAS® Event Stream Processing provides a framework and run-time architecture for building and deploying analytical models that run continuously on streams of incoming data, which can come from virtually any source: message queues, databases, files, TCP/IP sockets, and so on. SAS® Visual Scenario Designer is a powerful tool for developing, testing, and deploying aggregations, models, and rule sets that run in the SAS® Event Stream Processing Engine. In this paper, we will explore the technology architecture, data flow, tools, and methodologies that are required to build a solution based on SAS Visual Scenario Designer, enabling organizations to fight crime in real time.

INTRODUCTION

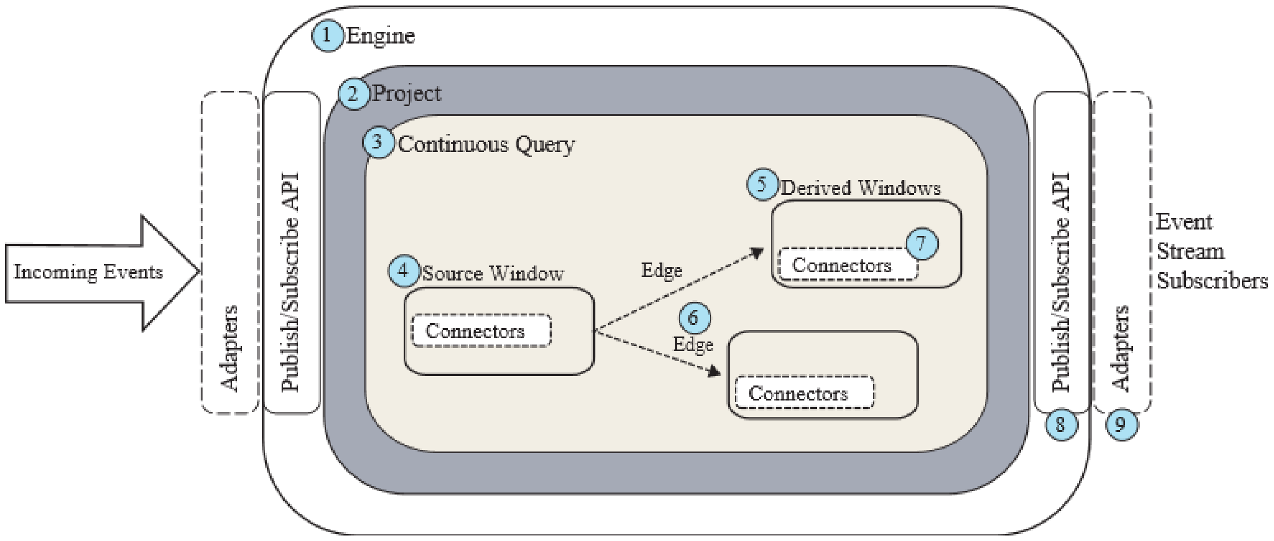
Traditional methods of performing fraud detection have relied on batch runs, periodically analyzing accumulated transactions to identify anomalous behavior.

As fraudsters and criminals have become more sophisticated, and are increasingly committing their crimes electronically, if we wait for the nightly batch run, the criminal might have already changed their identity, their Social Security number, their address, and their phone number by the time we have detected potential fraud.

To address this rapid pace of change within the criminal, SAS is offering real-time analytical capabilities, whereby potential fraud, identity theft, money laundering, and other crimes can be identified *at the time that the transactions occur.* This is the key paradigm shift in the analysis model for crime detection: batch versus real time.

SAS EVENT STREAM PROCESSING

SAS Event Stream Processing Engine provides this real-time analysis capability. SAS Event Stream Processing provides a generic capability for connecting to external data sources such as message queues, text files, databases, REST interfaces, and TCP/IP connections in order to stream data into continuous queries, as depicted below:



Display 1. Event Stream Processing, Architectural View

A continuous query is represented by a directed graph. This graph is a set of connected nodes that follow a direction down one or more parallel paths. Continuous queries are data flows, which are data transformations and analysis of incoming event streams.

Each query has a unique name and begins with one or more source windows.

Source windows are typically connected to one or more derived windows. Derived windows can detect patterns in the data, transform the data, aggregate the data, analyze the data, filter the data, or perform computations based on the data. They can be connected to other derived windows. Ultimately, the results of this event streaming execution can be pushed out to external systems via message queues, text files, databases, REST interfaces, and TCP/IP sockets.

FRONT RUNNING EXAMPLE

Front running provides a perfect use case for SAS Event Stream Processing in the arena of crime detection. Front running is the practice of a stock broker taking a large order for client, and executing a personal order for themselves *before* the large institutional order is executed. Once the large order is placed, the market adjusts to the newly demonstrated demand, the price of the stock increases, and the broker then sells his shares for a profit.

In order to detect front running, we would develop and deploy a SAS Event Stream Processing model based on a pattern window. A pattern defines events of interest that are associated with each other via logical operators, and which occur within various time-based conditions. For the front running case, the events of interest are as follows:

1) Event #1 (e1): Buy shares of a stock for yourself, Broker = Customer
2) Event #2 (e2) : Buying shares of a stock for a customer, Broker != Customer
3) Event #3 (e3) : Selling your stock for a profit, Sale > Purchase

Events of interest can be combined across events (rows) of the input event stream.

Various operators can be applied to combine events of interest. The “followed by” operator detects whether each event of interest is followed in the event stream by the one following it in the expression list. To detect the front running sequence, we would use the following operator:

```
fby(e1,e2,e3)
```

If we wanted to add a time element that would specify that all the events must occur within a certain time window, one hour, we could simply add:

```
fby{1 hour}(e1,e2,e3)
```

There are additional operators that could be used within pattern windows. If we wanted to detect the fact that a certain event had not happened within six hour time period, we could use the “does not occur” operator:

```
notoccur{6 hour}(e1)
```

If we wanted to detect the fact that a certain event happened immediately after another event, with no events occurring between the two, we could use the “is” operator.

```
e1 and is(e2)
```

The entire SAS Event Stream Processing XML for front running detection is listed below:

```
<window-pattern name='front_running'>
<schema-
string>id*:string,broker:int32,symbol:string,date1:string,date2:string,date3:
string,id1:int64,id2:int64,id3:int64</schema-string>
<patterns>
<pattern>
<events>
<event name='e1'>((buysellflg==’B’) and (broker == buyer) and (s == symbol)
and (b == broker) and (p == price)) </event>
<event name='e2'>((buysellflg==’B’) and (broker != buyer) and (s == symbol)
and (b == broker)) </event>
<event name='e3'><![CDATA[buysellflg==’S’) and (broker == seller) and (s ==
symbol) and (b == broker) and (p < price))]]></event>
</events>
<logic>fby(e1,e2,e3)</logic>
<output>
<field-selection name='broker' node='e1'/>
<field-selection name='symbol' node='e1'/>
<field-selection name='date' node='e1'/>
<field-selection name='date' node='e2’/>
<field-selection name='date' node='e3'/>
<field-selection name='id' node='e1'/>
<field-selection name='id' node='e2'/>
<field-selection name='id' node='e3'/>
</output>
</pattern>
</patterns>
</window-pattern>
```

TERRORISM SURVEILLANCE EXAMPLE

A common real-time pattern in identifying potential terrorist activity is in the area of flight ticket purchases. Purchases by cash or Visa gift card raise a flag. Long distance flights from a major metropolitan hub, to countries known for terrorist financing training, and back again in a short time window are suspect. This is a perfect real-time surveillance use case, as often the tickets are initially bought with a return of several days later, outside the window of suspicion, but are then later changed to a flight an hour from the time of purchase, the day after they arrived. If we waited to analyze all of the flight transactions in a nightly batch run, the individual would likely be back to the major metropolitan hub by the time the anomaly was found, armed with whatever they picked up on the trip. By analyzing the transactions *as they occur*, and sending alerts immediately to an investigation system such as SAS® Visual Investigator, authorities could be waiting for the suspects while they are waiting to get on the plane for their return flight home, and thwarting a potential attack. For this surveillance scenario, once again, a pattern window could be deployed that checks for a sequence of events as follows:

1) Event 1 (e1): An individual leaving from a list of identified airports (A), going to list of identified airports (B)
2) Event 2 (e2): The same individual booking a flight from (B) back to (A)
3) These two events occurring within a thirty six hour window.
    a. In the SAS Event Stream Processing model, use: `fby{36 hour} (e1, e2)`

AN ANALYST INTERFACE IS NEEDED

The contents of these streaming continuous queries, particularly those related to fraud detection and crime prevention, can get quite complicated. There are over a dozen windows types in SAS Event Stream Processing. Building SAS Event Stream Processing models, which are represented by an XML grammar, is much like programming. A visual tool for developing these XML models, SAS Event Stream Processing Studio, has been available, but the user of SAS Event Stream Processing Studio still worked at a very low level, wiring together dozens to hundreds of windows manually. For the fraud or security intelligence analyst, a higher level tool is needed to allow the user to codify and express higher level domain specific “scenarios”, and generate the appropriate SAS Event Stream Processing XML. This higher level tool is SAS Visual Scenario Designer.

SAS VISUAL SCENARIO DESIGNER – BACKGROUND AND EVOLUTION

SAS Visual Scenario Designer was originally released in 2014 on the SAS 9.4 platform, the SAS® LASR™ Analytic Server, and the SAS Event Stream Processing Engine. The purpose of SAS Visual Scenario Designer was to allow fraud and compliance analysts to explore and visualize their data, perform data-preparation operations, and author “scenarios” that detect anomalous behavior. Scenarios are encapsulated pieces of logic that determine whether a given account, customer, patient, doctor, or other entity represented in a data source is exhibiting behavior that warrants additional investigation. The canonical example of a scenario is “Alert on all the banking customers that have deposited over $10,000 in cash over the last seven days”. Another simple example would be, “Alert on all the physicians that are prescribing a drug or medical device 50% more than their peers in the same specialty.” SAS Visual Scenario Designer scenarios were developed, tested, and simulated exclusively against the SAS LASR Analytic Server, and could ultimately be deployed for production use into a SAS LASR Analytic Server environment or into SAS Event Stream Processing. Batch processing was typically performed in SAS LASR Analytic Server, and real-time/streaming processing was performed in SAS Event Stream Processing. While this provided a unified authoring and simulation environment between batch and real time, there were several differences between them that made testing and simulating against SAS LASR Analytic Server and deploying to SAS Event Stream Processing problematic:

- Time unit differences: SAS LASR Analytic Server supports WEEK/QUARTER/YEAR, SAS Event Stream Processing has no higher grain than DAY.

- Lookback behavior: Default SAS LASR Analytic Server behavior is that a DAY lookback goes from the transaction timestamp to the first tick of the current day. SAS Event Stream Processing behavior is that a DAY lookback goes 24 hours plus one tick from the transaction timestamp.
- Function differences: SAS LASR Analytic Server uses SAS functions. SAS Event Stream Processing uses DataFlux functions. The function signature and behavior of functions of the same category was not always identical.

One other key characteristic of SAS Visual Scenario Designer is that it relied on an overall alert management process outside of the solution. This overall process typically includes alert triaging, case management, case investigation, workflow, and alert dispositioning. When anomalous behavior was detected, and alerting events were generated, some other process or system needed to be developed or integrated with to handle those events.

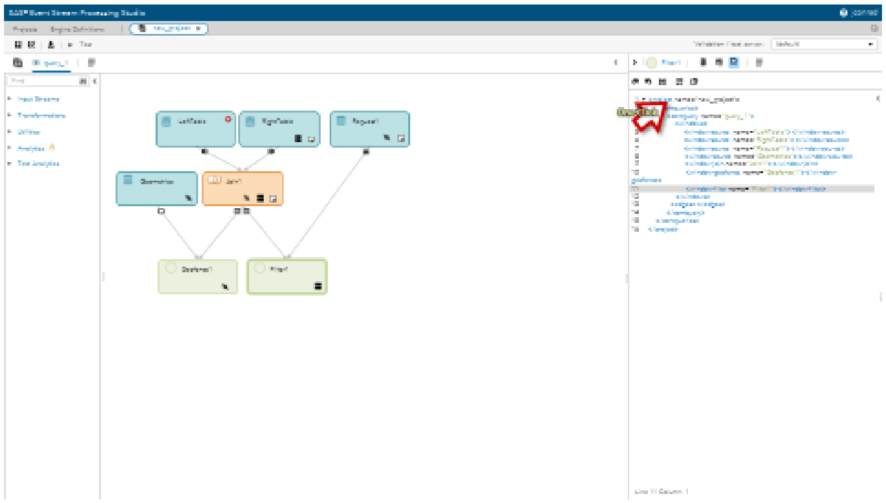
SAS VISUAL INVESTIGATOR –THE NEW SAS SURVEILLANCE AND INVESTIGATION PLATFORM

As SAS began looking toward to the future with SAS® Viya™ and SAS® Cloud Analytic Services (CAS), a turn-key surveillance and investigation system was envisaged, encompassing real-time surveillance, batch surveillance, data source management, alert triaging, case management, network and relationship analysis, workflow, and alert dispositioning. The 10.1 version of this new surveillance and investigation system, SAS Visual Investigator, was released in September 2016. The 10.2 version was released in March of 2017. A key design goal for SAS Visual Investigator was to provide all the elements of an alert generation and alert management process in one unified system. Another key design goal was to provide open interfaces for sending alerts to SAS Visual Investigator, such that a variety of surveillance approaches could be applied, each sending alerts to SAS Visual Investigator as needed. This open interface into the alerting system of SAS Visual Investigator was achieved via message queues and a well-defined alert document specification. Any system using the appropriate alert JSON format, and publishing alerts to the SAS Visual Investigator alert message queue, could participate in the SAS Visual Investigator surveillance infrastructure.

SAS Visual Scenario Designer underwent a significant re-architecture for the 10.2 SAS Viya release. Rather than try to author and test batch and real-time scenarios with a single interface, the batch functionality of SAS Visual Scenario Designer was re-branded into the Scenario Administrator component of SAS Visual Investigator. The real-time scenario authoring capabilities of SAS Visual Investigator were moved into SAS Event Stream Processing Studio, by way of a newly released extensibility capability, called the SAS Event Stream Processing Studio plug-in interface.

SAS EVENT STREAM PROCESSING STUDIO AND THE SAS VISUAL SCENARIO DESIGNER SURVEILLANCE PLUG-INS

With the 4.3 release of SAS Event Stream Processing Studio in March 2017, SAS Event Stream Processing Studio enabled other solutions to create higher level real-time constructs. A typical SAS Event Stream Processing Studio model is depicted below:



Display 2. SAS Event Stream Processing Studio Process Flow Diagram

Each node in the process flow diagram represents a low-level SAS Event Stream Processing window, source windows, a join window, a geo-fencing window, and a filter window. With the new plug-in, a new process flow diagram node could be created by a solutions team at SAS, by another software company, by a consultant in the field, by virtually anyone. Ultimately, these plug-in nodes generate a set of native SAS Event Stream Processing window primitives, but the end user doesn't need to know or care about the underlying window structure, they simply use the higher level construct, which comes with its own property pane. A surveillance "scenario" node might generate several source windows combined with a join window, a number of aggregation windows, and a filter window to execute the rule set. This plug-in capability allows SAS Event Stream Processing to be extended into virtually any domain, and will create a new plug-in market.

The first plug-in being developed by the SAS Visual Scenario Designer development team is for formatting an incoming stream into the correct JSON structure to be sent to the SAS Visual Investigator alert message queue. The key properties needed to send such an alert message to SAS Visual Investigator are as follows:

1) Entity Type. What are we investigating (CUSTOMER, ACCOUNT, POLICY, PHYSICIAN, PATIENT, and so on). These types come from the list of entities registered in the SAS Visual Investigator Data Hub.

2) Entity ID Field. For the incoming stream, what field represents the unique identifier of the Entity Type specified above.

3) Alert Queue. This is not to be confused with the RabbitMQ message queue. The Alert Queue is a routing construct within the SAS Visual Investigator alerting component.

4) Scenario ID: This is a unique identifier for the scenario that generated this alert. Each SAS Event Stream Processing surveillance model will be assigned a unique identifier so that the alerting module within SAS Visual Investigator knows how the alert was generated.

The SAS Visual Investigator alert plug-in will surface a properties pane with the four properties listed above, and will ultimately generate SAS Event Stream Processing procedure window XML, which creates a JSON string with the appropriate format to be ingested by SAS Visual Investigator. An example of this procedure window XML is below:

```
<window-functional name="Functional_1" pubsub="true" collapse-updates="true">
  <schema>
    <fields>
      <field name="sequence" type="int64" key="true"/>
      <field name="json_format" type="string"/>
    </fields>
  </schema>
  <function-context>
    <functions>
      <function name="json_format"><![CDATA[string('{
    "alertingEvents": [
        {
            "alertingEventId": "',$alertingEvents_alertingEventId,'",
            "actionableEntityType":
"',$alertingEvents_actionableEntityType,'",
            "actionableEntityId": ',$alertingEvents_actionableEntityId,',
            "score": ',$alertingEvents_score,',
            "alertOriginCd": "',$alertingEvents_alertOriginCd,'",
            "alertTypeCd": "',$alertingEvents_alertTypeCd,'",
            "alertTriggerTxt": "',$alertingEvents_alertTriggerTxt,'",
            "scenario_id": ',$alertingEvents_scenario_id,',
            "recQueueId": "',$alertingEvents_recQueueId,'"
        }
    ],
    "scenarioFiredEvents": [
        {
            "scenarioFiredEventId":
"',$scenarioFiredEvents_scenarioFiredEventId,'",
            "alertingEventId": "',$scenarioFiredEvents_alertingEventId,'",
            "displayTypeCd": "',$scenarioFiredEvents_displayTypeCd,'",
            "displayFlg": ',$scenarioFiredEvents_displayFlg,',
            "scenarioOriginCd": "',$scenarioFiredEvents_scenarioOriginCd,'",
            "scenarioDescriptionTxt":
"',$scenarioFiredEvents_scenarioDescriptionTxt,'",
            "scenarioId": "',$scenarioFiredEvents_scenarioId,'"
        }
    ],
    "enrichment": [
        {
            "alertingEventId": "',$enrichment_alertingEventId,'",
            "url": "',$enrichment_url,'",
            "date_published": "',$enrichment_date_published,'",
            "title": "',$enrichment_title,'"
        }
    ]
}')]]></function>
    </functions>
  </function-context>
</window-functional>
```

You can see that simply dragging a SAS Visual Investigator alert node on the SAS Event Stream Processing Studio canvas, and filling out a few properties, is far simpler than hand-rolling the above XML. The SAS Event Stream Processing Studio plug-in interface is a game changer for event stream processing at SAS.

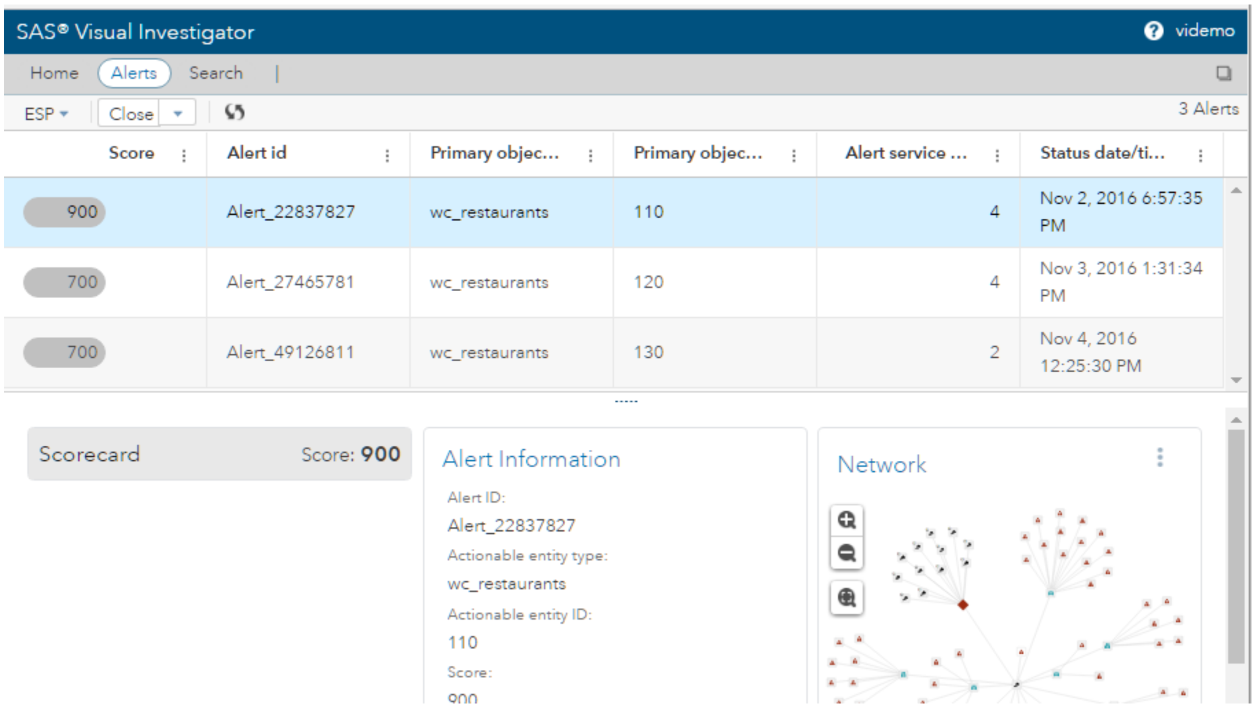
RABBIT MQ CONNECTOR

In order for the message created above to be sent to SAS Visual Investigator, it must be placed on the SAS Visual Investigator alert message queue. A SAS Event Stream Processing connector node can be configured as shown in the example below:

```
<connectors>
  <connector name="New_Connector_1" class="fs">
    <properties>
      <property name="type"><![CDATA[sub]]></property>
      <property name="snapshot"><![CDATA[true]]></property>
      <property name="fsname"><![CDATA[output.json]]></property>
      <property name="fstype"><![CDATA[json]]></property>
    </properties>
  </connector>
  <connector name="rmq_sub" class="rmq">
    <properties>
      <property name="type"><![CDATA[sub]]></property>
      <property name="rmquserid"><![CDATA[guest]]></property>
      <property name="rmqpassword"><![CDATA[guest]]></property>
      <property name="rmqhost"><![CDATA[10.38.13.246]]></property>
      <property name="rmqport"><![CDATA[5672]]></property>
      <property name="rmqexchange"><![CDATA[svi.tdc.exchange]]></property>
      <property name="rmqtopic"><![CDATA[svi.tdc.ae.q]]></property>
      <property name="rmqtype"><![CDATA[json_format]]></property>
      <property name="numbufferedmsgs"><![CDATA[50000]]></property>
      <property name="urlhostport"><![CDATA[esp-base:39908]]></property>
      <property name="buspersistence"><![CDATA[true]]></property>
      <property name="snapshot"><![CDATA[false]]></property>
      <property name="collapse"><![CDATA[true]]></property>
    </properties>
  </connector>
</connectors>
```

THE END GAME: ALERTS IN SAS VISUAL INVESTIGATOR

By using a Rabbit MQ Connector in our surveillance model, we are able to send alerts to SAS Visual Investigator from SAS Event Stream Processing. Below is a depiction of the incoming alert queue within SAS Visual Investigator. From this point, an investigator can raise/lower priority of the alert, can create a case for the alert, can initiate a workflow for the alert, can view all the entities related to this alert (customers, accounts, policies), and entities related to those entities, and so on, in a network visualization.



Score	Alert id	Primary objec...	Primary objec...	Alert service ...	Status date/ti...
900	Alert_22837827	wc_restaurants	110	4	Nov 2, 2016 6:57:35 PM
700	Alert_27465781	wc_restaurants	120	4	Nov 3, 2016 1:31:34 PM
700	Alert_49126811	wc_restaurants	130	2	Nov 4, 2016 12:25:30 PM

Scorecard Score: 900

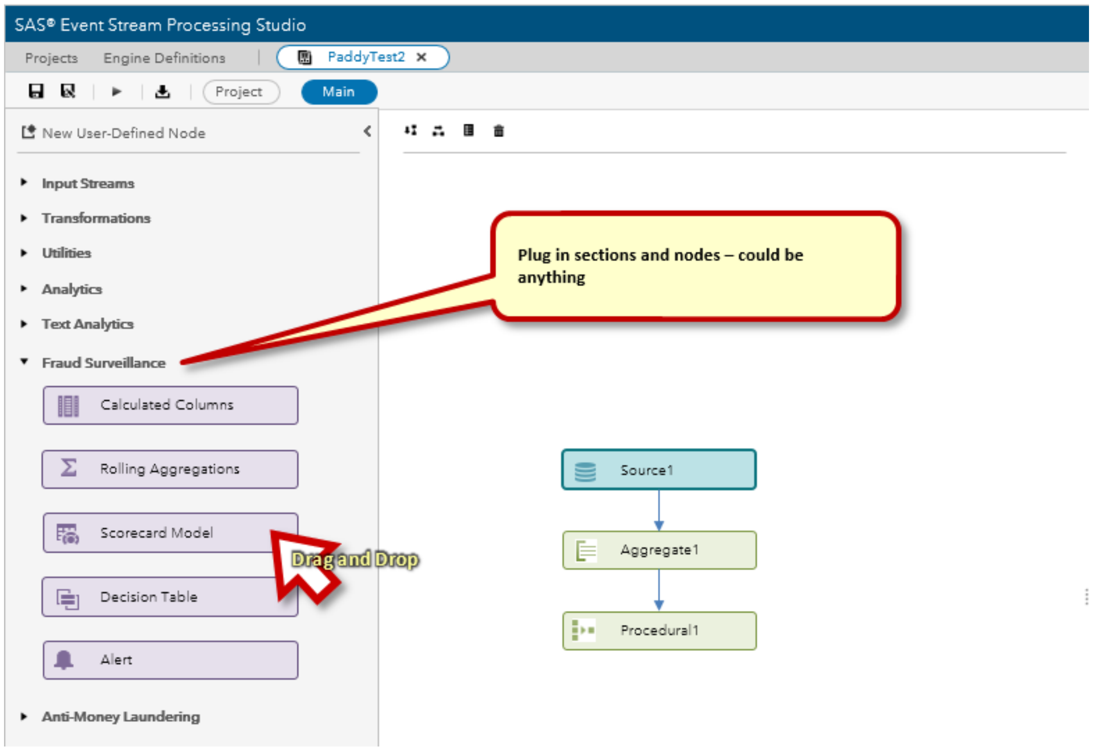
Alert Information

Alert ID:
Alert_22837827
Actionable entity type:
wc_restaurants
Actionable entity ID:
110
Score:
900

Display 3. Main Alert View in SAS Visual Investigator

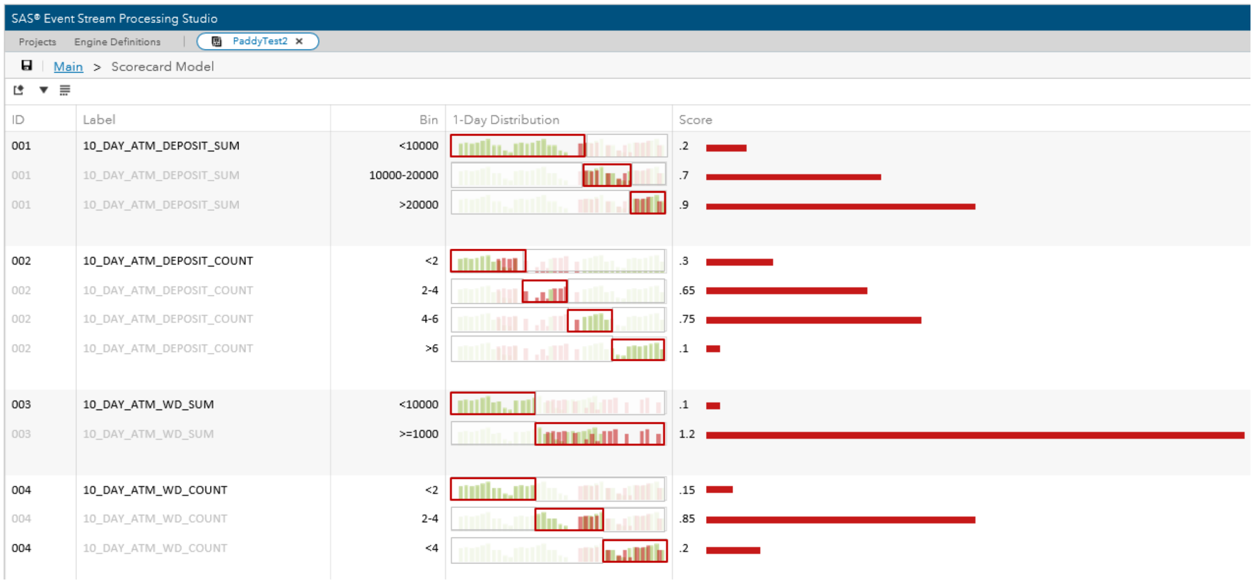
FUTURE PLUG INS

SAS is actively designing and developing a number of surveillance plug-ins for SAS Event Stream Processing Studio. We envision that various solution teams will create entire plug-in sections that can be surfaced through the process flow diagram pallet, as depicted below:



Display 4. SAS Event Stream Processing Studio Plug-in Extensions List

Two powerful constructs used in surveillance scenarios are decision tables and scorecard models. Decision tables allow for a highly flexible if, then, else, else, else heuristic rule creation. Scorecard models allow multiple aggregations to be considered jointly, with fine tuning of lookback time periods (1 DAY, 10 SECONDS, 5 MINUTES), aggregation functions (SUM, COUNT, MEAN, STANDARD DEVIATION, and so on). A design for a forthcoming scorecard model plug-in is depicted below:



ID	Label	Bin	1-Day Distribution	Score
001	10_DAY_ATM_DEPOSIT_SUM	<10000		.2
001	10_DAY_ATM_DEPOSIT_SUM	10000-20000		.7
001	10_DAY_ATM_DEPOSIT_SUM	>20000		.9
002	10_DAY_ATM_DEPOSIT_COUNT	<2		.3
002	10_DAY_ATM_DEPOSIT_COUNT	2-4		.65
002	10_DAY_ATM_DEPOSIT_COUNT	4-6		.75
002	10_DAY_ATM_DEPOSIT_COUNT	>6		.1
003	10_DAY_ATM_WD_SUM	<10000		.1
003	10_DAY_ATM_WD_SUM	>=1000		1.2
004	10_DAY_ATM_WD_COUNT	<2		.15
004	10_DAY_ATM_WD_COUNT	2-4		.85
004	10_DAY_ATM_WD_COUNT	<4		.2

Display 5. Scorecard Model Plug-in

ADVANCED ANALYTICS IN REAL TIME

The preceding examples use relatively straightforward event sequence analysis, aggregations, and rule filtering. The SAS Event Stream Processing Engine can handle far more advanced analytics in order to surveil incoming data streams. Analytical models built with SAS® Enterprise Miner™ or SAS® Visual Statistics, and exported as DATA step or DS2 code can be used in SAS Event Stream Processing via a procedure window. Neural networks, decision trees, logistic regression, and linear regression models can all be resolved to DATA step code that SAS Event Stream Processing can use via a procedure window. Further, analytical models developed in Python can also be natively embedded into SAS Event Stream Processing models. Integration with SAS® Model Manager for model tuning, model comparison, and model deployment is in the works. The full power of the SAS 40+ year history in advanced analytics can be used in streaming, real-time models.

CONCLUSION

Batch analysis of data and periodic alert generation has been the standard for financial crimes and security intelligence investigations for a number of years. With the proliferation of the Internet, online banking, identity theft, hacking, phishing, and electronic crime, the modern criminal has often already erased their footprints, destroyed their fraudulent identity, and electronically moved on to their next identity by the time that the batch analysis has completed and human investigators have been alerted to suspicious behavior. In the age of the digital criminal, real-time analysis and alert generation, occurring at the time that events are occurring, is proving to be a far more effective method of crime detection and prevention. SAS has an event stream processing engine for such real-time analysis detection. With the advent of the SAS Event Stream Processing Studio plug-in interface and the forthcoming SAS Visual Scenario Designer surveillance plug-ins, SAS is empowering organizations to take surveillance authoring to the next level and to fight crime in real time.

REFERENCES

SAS Institute Inc 2016. Help for SAS Event Stream Processing 4.2. Cary, NC: SAS Institute Inc. http://go.documentation.sas.com/?cdcId=espcdc&cdcVersion=4.2&docsetId=espov&docsetTarget=home.htm&locale=en

SAS Institute Inc. SAS Event Stream Processing 4.2 Training Manual. Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

Paddy Fahey, Senior User Experience Designer, SAS Institute Inc. for his help in understanding Pattern Windows, and his design content on plug-in sections within SAS Event Stream Studio and the Scorecard model plug-in.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author:

John Shipway
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
john.shipway@sas.com
http://www.sas.com