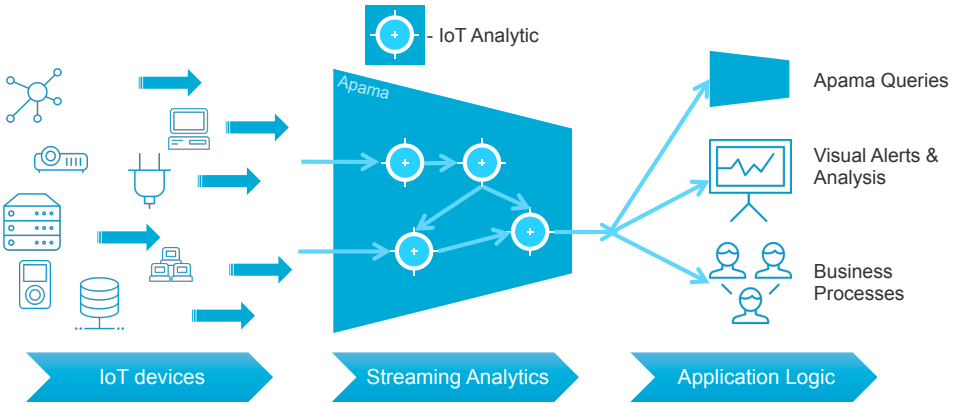
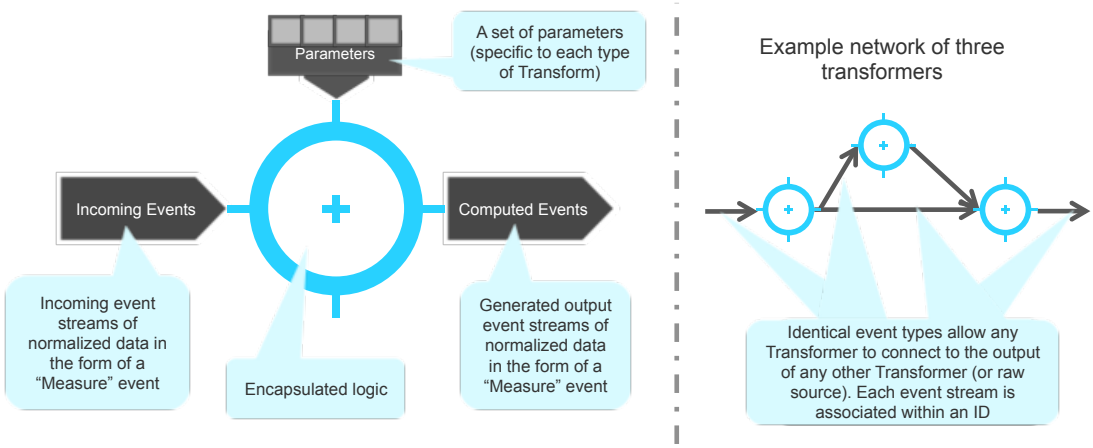
IoT Analytics Kit

Overview

The IoT Analytics Toolkit consists of a set of re-usable and scalable “micro services” that perform a range of streaming analytics and transformations over event streams. The scope of the analytics within the Toolkit is that they consume and generate defined “Measure” events. The elements within the IoT Analytics Kit are referred to here as “Transformers”.



Transformers have to have the ability to listen for Measures event so that they can use incoming raw data, and the ability to send Measure events to other Transformers or send them externally so that Apama Queries and other component can be used if required. In this way streaming networks of analytics can be defined.



Most Transformers are implemented such that they automatically multiplex the incoming data streams; this means that all data values for all sensors on a single stream fed into a single Transform, e.g. Moving Average, will incur separate (moving average) calculations for each individual sensor ID, and each output from that Transform will be the updated moving average for that specific sensor. So, a single instantiated Transform can perform the required calculation for any number of individual

sensors on a single stream. Should sensor values need to be combined, then an Aggregator transform may be used in front of the required calculation Transform(s).

Apama Event Structures

A "Measure" event is a normalized event structure that will be used within the toolkit for passing around values (both from sensors and Transformers). Events coming in from external sources will need to be converted into this event format prior to use:

```
event Measure {
   string    name;                 The name of the measurement stream
   string    type;                 Enumerated type(Measure Types) describing
                                   this event as ANOMALY, COMPUTED or RAW
   string    assetId;              The id of the source (normally a sensor)
   decimal   timestamp;            The timestamp of the measurement
   decimal   dValue;               The value of the measurement as a decimal
   string    sValue;               The value of the measurement as a string
   float     xValue;               Three floats intended for supplementing data
                                   in dValue or sValue (such as a location)
   float     yValue;               //
   float     zValue;               //
   dictionary <string, string>     To hold any other data associated with the
                 extraParams;      measurement
}
```

The intent is to have a set of analytics (Transformers) that can be chained together as desired. They are created though the generation of a single event type. The structure of this "Transform" event type is:

```
event Transform {
   string name;                                  The name of this new node
   sequence <string> inputMeasureNames;          The stream(s) to receive
   sequence <string> outputMeasureNames;         Names of the stream(s) to
                                                 generate
   dictionary <string, string> params;           A list of parameters to customize
                                                 this instance
}
```

For example:

```
Transform("MovingAverage", ["Input"] , ["1MinMA"], {"timeWindow":"60.0d"})
```

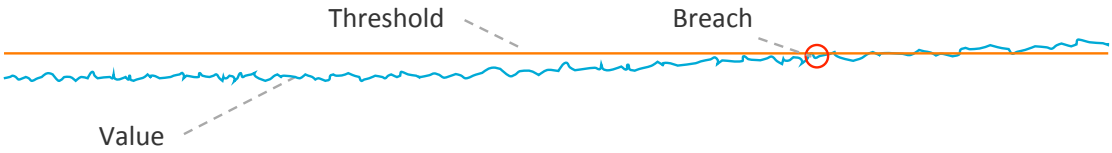
This calculates a 60 second Moving Average of "Input" feed and generates a "1MinMA" feed, also:

```
Transform("Variance", ["1MinMA"], ["Vols"] , {"timeWindow":"3600.0d"})
```

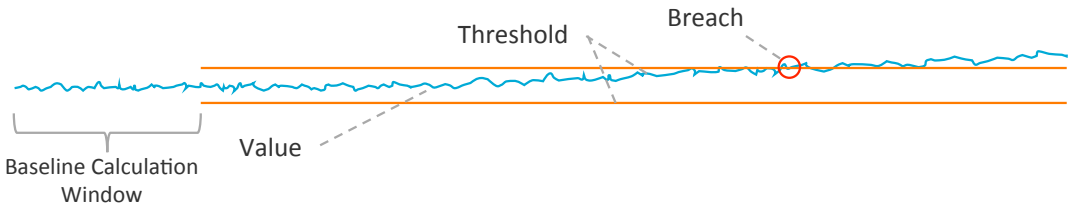
Will calculate a 1 hour Variance (volatility) calculation over the earlier calculated moving average feed (from 1MinMA), and distribute this on the "Vols" feed for further consumption.

Example Transformers

The ThresholdBreach Transform will generate alerts when a data value from the incoming data stream exceeds a given value (in a direction you specify):



An extension of this is the BaselineThreshold, which will automatically compute the threshold value from the initial set of readings from the sensor rather this being required to be manually entered. It then looks for deviations from this value. The assumption is that in the initial conditions, the sensor is behaving normally. This can be used to ensure that sensor values don't slowly drift from initial conditions:



The Nominal Range Transformer will continually use the moving average and continuous volatility calculations to define an acceptable range for a sensor value that will be dynamically and continuously computed. Only when the value exceeds one of these thresholds will an alert be generated. This Transformer can usefully be used to automatically define a "nominal" operating range for a sensor without prior knowledge of what that range is and where the value itself can correctly slowly move around a wide value range.

Technical Notes

- Timestamp Values are in Apama standard format: Float value of **seconds** since 1970 [Java standard is milliseconds since 1970]
- Transformer creation event is intentionally made uniform to allow simple creation of transformer networks through scripts or a graphical editor

Included Sample

The sample within the IoT Analytics Kit gives a simple end-to-end demonstration of Transformers in use and includes a simple simulator to generate data feeds and an Apama Dashboard to visualize the results.

The Sample:

- Includes a Nominal Range Transformer to identify when each individual sensor's value is within "normal" operating bounds **(1)**. Also allows the value of one sensor to be manually altered **(2)** to see the impact of large changes using the slide on the upper right hand side. By moving this you can modify the simulated value for that sensor manually (e.g. to simulate a dramatic change)
    - Use the drop-down box **(3)** at the top of the user interface to browse through the details (ie: the data calculated by the Nominal Range Transform) for each individual sensor.

- Calculates the gradient **(4)** of recent values for each sensor, so we can tell how much a sensors value is changing.
    - The gradient of the selected AssetID is also shown on a dial **(5)**.
- Exposes a chart of the current values of all sensors **(6)**
- Calculates the moving average of each sensor in the chart at the bottom **(7)**.