REST Business Activity Monitoring (rBAM)

V1.0 (Jan 2020)

Author: Steve Pisani

Jan 2020

Table of Contents

Overview:

This code extends on the functionality provided by IRIS Business Metric classes as follows

Assuming an already existing, and running Business Metric class:

1) Collect and push (using HTTP POST) the metric values to a nominated REST endpoint. This is useful if you want to distribute metrics up to a remote system. For example – using this feature one can push the metric values as a Power BI real-time DataSet which can be then consumed by Microsoft PowerBI Dashboards for real-time visualization in that framework.

2) Collect and host (using a REST API) the metric values that external systems invoke in order to retrieve metric data. The API provides:
   a. The list of business metric classes running in a production
   b. JSON for the metric values of a given enabled Business Metric Class
   c. JSON for the metric values of all enabled Business Metric Classes

It is worth mentioning that this functionality is accomplished without needing to modify, subclass or otherwise extend any existing Business Metric class code you have. This functionality taps into the tables currently defined by IRIS, that hold the most recent calculated metric values.

It is possible to import the entire package and implement either or both of the options offered here.

Installation:

- Clone or download/extract the repository into an empty folder on your system
https://github.com/pisani/REST-BusinessActivityMonitoring

- Open an InterSystems IRIS Terminal Window

- Switch into a Namespace configured for interoperability.

- Import the code into your namespace by executing execute the following command, where
<yourTempDir> is the folder containing extracted repository.(Note this imports the entire package) :

> **do $System.OBJ.LoadDir(<yourTempDir>,"ck",,1)**

Note – The code above imports the complete set of classes, allowing you to HTTP POST or host (via a REST API) of the metric data. Additionally, sample Metric and Production classes are provided.

Selection, based on the table below allows you to refine which classes you retain. For example, if you just want to PUSH data, (and do not need care for the sample classes), you only require the **zaux.rBAM.Utils, and, zaux.rBAM.Operation** classes. (the zaux.rBAM.API* and zaux.rBAM.Sample* classes may be removed).

See below for Package content and each class's individual purpose:

Package Contents

Item	Purpose	Usage
zaux.rBAM.Utils.cls	Core utilities class	**Yes – Always Required.**
zaux.rBAM.Operation.cls	Capture and PUSH metric values to external server	Only if intending to post metric data to external REST endpoint.
zaux.rBAM.API.cls	REST Dispatch class to host REST Endpoint for consumers	Only if hosting REST API for external REST clients to consume metrics
zaux.rBAM.API.v1.cls	Version 1 implementation of REST API	Only if hosting REST API for external REST clients to consume metrics
zaux.rBAM.Sample.*	Sample Interoperability Production and Sample Metric Class generating random data	Optional used only for demonstration purposes.

HTTP POST Business Operation for sending metric data to an external endpoint

1. Add the Business Operation **zaux.rBAM.Operation** as a new operation into your integration production, ensuring you enable it.
2. Under the Settings Category labeled **Basic Settings** specify the intended REST endpoint's details – as a minimum: HTTP Server, HTTP Port and URL. Specify any Basic Authentication security credentials here too.
3. If using HTTPS, specify SSL Configuration which you can define via the System Management's Administration section of your instance.
4. Under the Setting Category labelled **REST Output** specify configuration settings that control the interval and format of the POSTed JSON data:

Call Interval: How often (in seconds) the business operation will check for record metric values, and issue an HTTP POST to the defined end point.

Set to ZERO (0), if manually invoking the operation to post on an ad-hoc basis, triggered only when the operation is sent a message. To do so, send the Business Operation an instance of Ens.StringRequest with **Value** property equal to a comma separated list of Business Metric configuration names. This mode overrides the next setting 'ServiceClassNames'.

ServiceClassNames Used only on an automatic cycle, select all Business Service Metric classes that would be referenced for output of their metric properties, via this business operation

StructuredOoutput Structured, or unstructured formatted JSON output.
See Appendix: **Configurable output parameters** for details.

SkipEmptyJSON If TRUE, and JSON data is empty, do not make an HTTP Request.

ClassNamePrefix For when StructuredOutput is FALSE only- choose to explicitly include metric class names in JSON attribute names,
See Appendix: **Configurable output parameters** for details.

HideMetrics Populate to filter out nominated metrics
See Appendix: **Configurable output parameters** for details.

MetricParameters Set to TRUE to expose metric parameters (RANGEUPPER, RANGELOWER,. Etc)
See Appendix: **Configurable output parameters** for details.

PathForDebugFile Optional file path to receive the debug output file 'JSONPostData.txt'

Building a Power BI Dashboard from IRIS Metrics by populating a Microsoft Cloud-based Streaming Dataset.

You may publish IRIS metrics as a Streaming Dataset to Microsoft's Power BI in order consume these values in constantly updating Power BI Dashboards.

Note that IRIS also has the capability to consume and display Business Metrics natively, but a wider range of graphical widgets to choose from (see: Business Activity Monitoring: https://docs.intersystems.com/irisforhealth20194/csp/docbook/DocBook.UI.Page.cls?KEY=EGIN_options#EGIN_options_bam).

However – whilst very basic in its visualization controls, real-time Power BI dashboards can incorporate data from multiple streams (hence multiple IRIS productions, or other sources easily), and readily formats the dashboard for mobile devices.

Warning: Data pushed to the cloud this way ends up going through infrastructure outside of your organization, therefore – this may raise security concerns and do not publish sensitive data.

In this example, we will use the provided **zaux.rBAM.Sample.Production**, and **zaux.rBAM.Sample.MetricClass** (which generates random numbers for metric values) as a data source.

Setup Power BI Streaming Dataset

1. Log into your Power BI Microsoft account: powerbi.microsoft.com

2. With your **Workspace** selected, refer to the top right-hand drop menu **+ Create**, and select Create -> Streaming Dataset
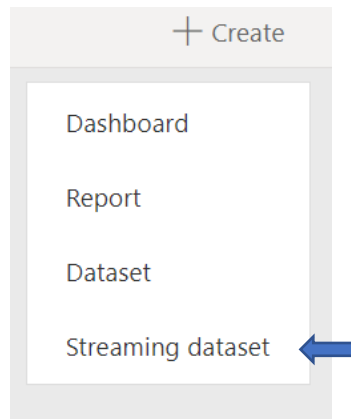
3. Select **{ API }** to define a generic streaming dataset, then click NEXT.

4. Give your Dataset a name (eg: 'IRISStreamingDataset'), and add the values:

   _Production   as Text
   _SampleDateTime   as DateTime
   AverageTemp   as Number *

   * 'AverageTemp is the only metric we are going to publish from the IRIS production.

   'Historic data analysis' can be selected if you want Microsoft to accumulate more than a handful of readings over time. This can be useful if building a Power BI Report (not Dashboard) and needing to analyse.

   The screenshot below shows a populated Streaming Dataset populated. Select CREATE to save this Streaming Dataset definition

* Required

Dataset name *

IRISStreamingDataset

Values from stream *

_Production   Text   ▼ 🗑

_SampleDateTime   DateTime   ▼ 🗑

AverageTemp   Number   ▼ 🗑

Enter a new value name   Text   ▼

[
  {
    "_Production" : "AAAAA555555",
    "_SampleDateTime" : "2020-01-23T03:20:52.509Z",
    "AverageTemp" : 98.6
  }
]

Historic data analysis

(●　) Off

5. Once created, Power BI will determine and provide a URL to use to supply data. In this case that will be done by IRIS.

    Copy the Push URL provided, as we need to supply this to the IRIS Business Operation
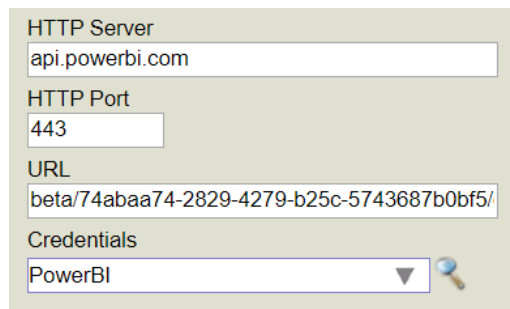
Push URL

https://api.powerbi.com/beta/74abaa74-2829-4279-b25c-5743687b0bf5/dat   ⬅

Raw   cURL   PowerShell

[
  {
    "_Production" : "AAAAA555555",
    "_SampleDateTime" : "2020-01-23T03:21:42.982Z",
    "AverageTemp" : 98.6
  }
]

8

Setup IRIS Operation

1. If you have not done so already, import the zaux.rBAM.Sample classes into an Interoperability enabled IRIS namespace, and start the zaux.rBAM.Sample.Production

2. Notice the Business Operation that will make the REST call in the production: zaux.rBAM.Operation.

3. Define a set of credentials containing your Microsoft username and password for Power BI.

4. Under Basic Settings, configure HTTP Server, HTTP Port and URL – with elements from the POST URL provided by Microsoft Stream Dataset properties, as well as the Credentials name you chose to use : eg:

Note: URL is everything after 'api.powerbi.com' in the POST URL provided to you.

5. Specify an SSL Certificate for the communication between IRIS and Microsoft in the SSL Configuration field.

6. Configure the business operation with settings for this example :
   StructuredOutput   – **Unchecked** (Important !)
   MetricParameters   – Unchecked in this example case
   ServiceClassName   – Note this is already selected for you as zaux.rBAM.Sample.MetricClass

7. Some metrics have been hidden using the configuration setting HideMetrics, where certain metric values are note distributed. The specified string:

   ::CO2Level,::CurrentHumidity

   hides the metric 'CO2Level' and 'CurrentHumidity', regardless of Configuration Name or Metric Instance, leaving just the metric "**AirQuality**" to be published to the streaming dataset.

8. Startup the **zaux.rBAM.SampleProduction** to start posting Metric data to the nominated endpoint.

9. Optionally – for debugging purposes, specify a host file path in setting 'PathForDebugFile' that will receive an output text file 'JSONPostData.txt' showing the most recent values that will be posted with every HTTP call.

Creating the Power BI real-time Dashboard

1. Return into your PowerBI Microsoft account: powerbi.microsoft.com
2. With your **Workspace** selected, refer to the top right-hand drop menu **+ Create**, and select Create -> Dashboard
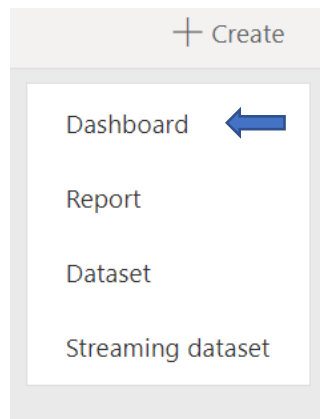
3. Provide a name for your dashboard, eg: "IRISMetricsDashboard"

4. Using the drop down menu activated by selecting elipse (…) in the menu ribbon
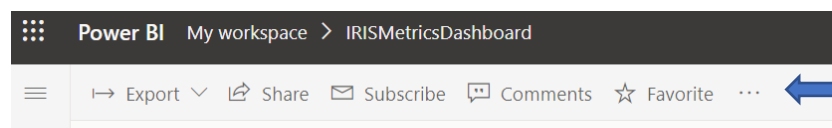

   And the option **Add Tile** to add the first tile to your dashboard.

5. Select **Custom Streaming Data** and click **Next**

6. Locate the Streaming Dataset "IRISStreamingDataset" previously defined and click **Next**

7. At this point select the visualization to use. You have a limited choice of a Card, Line Chart, Gauge or Clustered Bar/Column chart. For this example select **Line Chart**

8. Now all that is left to do is supply the following parameters:

   For **Axis**:   _SampledDateTime
   For **Values**:   AverageTemp

   Once these values have been set (and even before selecting NEXT) – as long as the IRIS production is correctly configured, and started – you should see values being plotted on your tile:

Line Chart populated from streaming dataset that's receiving values from an IRIS Business Metric
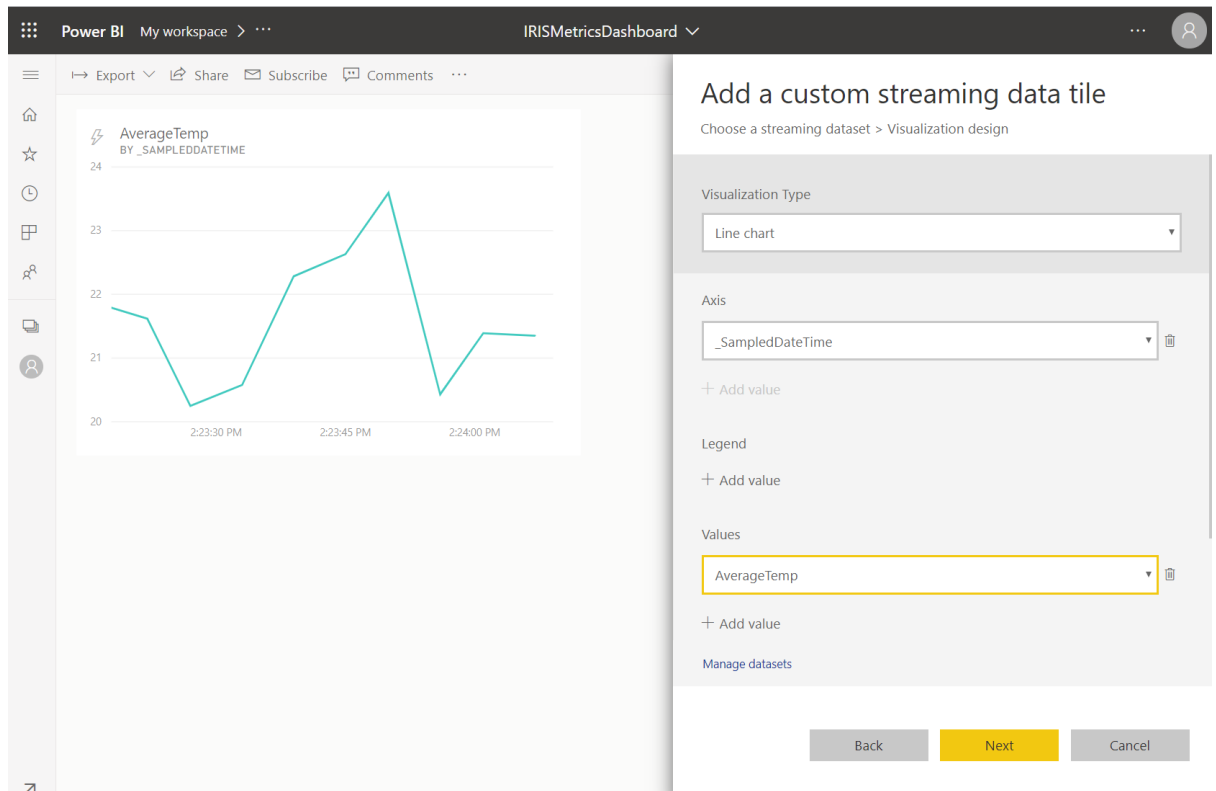
9. Click **Next** and modify the tile's Title, Subtitle and other parameters if needed, and to finalise this tile.

10. At this point you may continue to add more tiles, with other visualisations, and populated from the same or other IRIS data stream – or additionally, publicly published data streams.

REST API service

Hosting a REST API for providing business metric data to external REST Clients:

1. **Define a WEB Application** with appropriate dispatch class.

    -   Create a WEB Application and REST Endpoint to be used by external REST Clients. Using IRIS's System management portal navigate to:
        System > Security Management > Web Applications
    -   Click on **Create New Application**
    -   For **Name:** Specify the URL you want to for this endpoint. Below I have used "/metrics"
    -   For **Namespace**: Specify IRIS namespace where zaux classes have been loaded.
    -   Enable **REST** and for Dispatch Class specify: **zaux.rBAM.API**

1. Returns a list of enabled Business Metric Classes in the running production:

   /metrics/v1/List

```json
{
    "_Production": "zaux.rBAM.Sample.Production",
    "items": [
        {
            "ConfigName": "EnvironmentMetrics",
            "DataURL": "/metrics/v1/Data/EnvironmentMetrics",
            "Enabled": 1
        },
        {
            "ConfigName": "InventoryMetrics",
            "DataURL": "/metrics/v1/Data/InventoryMetrics",
            "Enabled": 1
        }
    ]
}
```

2. Returns metric data for all Business Metric Classes items enabled in the running production: (using the default output parameters)

   /metrics/v1/Data

```json
[
    {
        "_Production": "zaux.rBAM.Sample.Production",
        "_SampledDateTime": "2019-12-17 11:06:05",
        "EnvironmentMetrics_AirQuality_CarPark": 205,
        "EnvironmentMetrics_AirQuality_LoadingDocks": 104,
        "EnvironmentMetrics_AirQuality_Offices": 302,
        "EnvironmentMetrics_AverageDailyTemp": 27,
        "InventoryMetrics_OrdersReceived": 25,
        "InventoryMetrics_OrdersFilled": 15,
    }
]
```

3. Returns the metric data for a given Metric class if enabled:

   /metrics/v1/Data/*itemName*

Where *ItemName* is the configuration name of a Business Metric class

Appendix: Configurable output parameters

The following settings help with the format of the JSON document that is produced either by the Business Operation when sending metric data, or, by the REST API when returning data in response to an HTTP GET from a REST Client.

When utilizing the Business Operation – these settings can be found under the Operation's **REST Output** settings.

When invoking the REST API, these settings are defined as URL Parameters:

StructuredOoutput	Set to 0 if the JSON output is a flat (non-hierarchical) list of JSON attributes without arrays. This is the preferred format for Micrososft Streaming Datasets.
	Set to 1 if the metric properties, instances and metric class names are going to be output as a structured, hierarchical JSON body where metrics are a collection of attributes to Business Metric Instance, which in turn is a collection for each business metric Class enabled.
ClassNamePrefix	Applicable only when StructuredOutput is FALSE, this will determine whether the Business metric ClassName is included as a prefix in the JSON attribute for each metric property broadcast.
HideMetrics	This settings allows you to hide the output of specific business metric class data, instances, individual metrics, or a combination thereof. This settings takes a comma delimited string for each combination to hide. Default is to include all.
	For example:
	"::AirQuality" hides any metric labelled 'AirQuality', from any Instance or Metric Class, from output
	":North:AirQuality" hides any metric labelled 'AirQuality', from the I nstance "North" gathered under any Metric Class, from output
	"MyMetricCLass::AirQuality" hides any metric labelled 'AirQuality', regardless if the instance name gathered under any Metric Class "MyMetricClass", from output
MetricParameters	This setting allows you to include UNIT ,RANGEUPPER/LOWER and THRESHOLDUPPER/LOWER values for each business metric included. Default is FALSE.

Potential Future Enhancements

1. When specifying the HideMetrics configuration setting, allow the use of wildcards.