Title: Arbitrary SQL Monitor for AppDynamics

**Arbitrary SQL Monitor for AppDynamics**

An AppDynamics Machine Agent extension to run SQL statements against a JDBC database, and import the results of the query as custom metrics in AppDynamics.

Sample use cases:

* Import business metrics and KPI's from an application database.
* Import performance data from SolarWinds or other system monitors that use an RDBMS backend.

Key Features:

* Support for multi-row resultsets – Design the SQL query such that each output row is in a “name value pair” format – first field is name of metric and second field is value of metric
* Caching of data – Extension should cache result data and only query database only when cache expires. Cache configuration period is configurable through the property servers🡪 cacheTimeout in config.yml
* Support for no duplicate data – If the queried table(s) have a timestamp column with timestamp of when row was created, the extension be configured to process data only once.

This extension requires the Java Machine Agent.

**Installation**

* Download ArbitrarySQLMonitor.zip from the Community site.
* Copy ArbitrarySQLMonitor.zip into the directory where you installed the machine agent, under $AGENT\_HOME/monitors.
* Unzip the file. This will create a new directory called ArbitrarySQLMonitor.
* In $AGENT\_HOME/monitors/ArbitrarySQLMonitor, edit the file monitor.xml and configure the plugin.
* Copy your JDBC driver jarfile into $AGENT\_HOME/monitors/ArbitrarySQLMonitor/lib. The extension is pre-configured with an MS SQL driver.
* Restart the machine agent.

**Configuration**

Configuration for this monitor is in the monitor.xml and yml.config files in the monitor directory.

Note-Please make sure to not use tab (\t) while editing yaml files. You may want to validate the yaml file using a yaml validator **http://yamllint.com/**

1. Configure the database server instances by editing the config.yaml file in `<MACHINE\_AGENT\_HOME>/monitors/SQLMonitor/`. Below is the format

* server - The database server name
* driver - The class name of the JDBC driver to use e.g. com.mysql.jdbc.Driver
* connectionString - The connection URL for the driver e.g. jdbc:mysql://localhost:3306/demo
* user - The user name to connect as
* password - The password for the user. Encrypted password is supprted

metricPrefix: "Custom Metrics|SQL|"

2. Configure the path to the config.yaml file by editing the <task-arguments> in the monitor.xml file. Below is the sample

<task-arguments>          <!-- config file-->            <argument name="config-file" is-required="true" default-value="monitors/SQLMonitor/config.yml"     />          .... </task-arguments>

Set the below argument name for "timestamp-file" to the relative path of the extension folder location. For example:

<argument name="timestamp-file" is-required="false" default-value="monitors/ArbitrarySQLMonitor/timestamp.txt"/>

Set the time in timeStamp.txt located in the ArbitrarySqlMonitor folder to the most current date/time in this format:

2015-07-06T20:20:10.777-05:00

Be specific to the minute when re-configuring. This file is written to with the date/time of execution of the queries each time they run. If duplicate data is detected, this date/time in the file and the current date/time are subtracted from each other and the difference is passed into the queries where freqInSec is specified in the queries for a single cycle.

Go to monitor.xml to the 'task-arguments' tag and find this tag within it:

argument name="execution\_freq\_in\_secs"

\*You must set this field equal to whatever value is configured for the 'execution-frequency-in-seconds' field within the same file.

Next set

argument name="timeper\_in\_sec"

This is the value, in seconds, that will replace the variable

freqInSec

within the SQL queries in config.yml.

\*Note: ALWAYS set "timeper\_in\_sec" LESS THAN "execution\_freq\_in\_secs"

This ensures no duplicate data.

For example, you would put "freqInSec" in each SQL query to replace the time, in seconds, that you wish to run the query for:

DATEADD(ss, -freqInSec, GETDATE()))

Note: If freqInSec is left out of the SQL queries in config.yml, the extension will run as designed and simply not modify any queries nor validate for duplicate data.

**JDBC Driver**

To use this extension, you will need to provide the JDBC driver, class name, and connection URL. We've provided examples for some of the common databases. You'll need to replace the placeholders (HOST, PORT, DB, etc.) in the URL with your own values.

**Metrics Provided**

The metrics created by this extension depend on the query you provide.

* If the output of query is a single column, the column name will be the metric name
* If the output of a query is two columns, the first column name will be used for metric name and second column value will be the metric value
* Multiple result sets are supported.

**Support**

For any questions or feature requests, please contact the [AppDynamics Center of Excellence](mailto:ace-request@appdynamics.com).

**Version:** 1.0  
**Controller Compatibility:** 3.6 or later  
**Last Updated:** 10-June-2015  
**Author:** Aaron Jacobs

**Contributing**

Always feel free to fork and contribute any changes directly via [GitHub](https://github.com/Appdynamics/arbitrary-sql-monitoring-extension).

**Community**

Find out more at our [Community](http://community.appdynamics.com/) site.

**Release Notes**

**Version 1.0**

* Initial release.