# REST Query

This project lets you make simple REST API calls (GET, POST, etc.) to JSON web services, from custom actions in Resilient.

## Project Overview

This integration listens for messages on an Action Module message queue. When it receives a message – triggered by an Automatic Rule or a Menu Item Rule in Resilient – it performs a HTTP REST call according to a “query definition” that has been configured on the server. The query definition is a template that provides static or calculated values for

* The URL,
* The method (GET, POST, etc)
* Any custom HTTP headers to set
* Any JSON body for a POST request
* Instructions for processing the response to update Resilient with the result.

For example, you could use this to build actions that

* Call GET to a CMDB service where part of the URL is the value of the artifact that triggered the custom action. Read a list of values from the result, and create or update data-table rows from these values.
* Call GET where part of the URL is composed of fields from the incident. Put the result into a new Note.
* Call POST to send to a ticketing system from a task, and update the task name with the ticket ID from the result.
* Call POST to send an IP address to a firewall, and update the artifact description with the action taken.

## Installation

The project is delivered as a ‘pip’-installable resilient-circuits component. It requires ‘query-runner’. So the complete set of components to install will be

co3-27.1.22.tar.gz

resilient\_circuits-27.1.22.tar.gz

rc-query-runner-27.0.0.tar.gz

rc-query-rest-27.0.0.tar.gz

All these can be installed at once with

sudo pip install rc-query-rest-27.0.0.tar.gz --find-links .

Once installed, it runs as part of your resilient-circuits application,

resilient-circuits run

Also it will show up in the list,

resilient-circuits list

## Configuration

You need to configure the [rest] section of your app.config. To insert a template version of this into an existing ‘app.config’ file,

resilient-circuits config –u app.config

The config parameters are,

[rest]

queue=rest

# Directory containing query definition files

# Filenames must match the Resilient action names

query\_definitions\_dir=/usr/share/integration/queries\_rest

query\_timeout=60

After editing the config parameters, you will need to restart the ‘resilient-circuits’ process.

## Query Definition Files

Query definition files are stored in the ‘query\_definitions\_dir’ specified in config. When an action is triggered, the integration searches for a file with the same name as the action. Action names are lowercased, and non-word characters replaced with underscore. So a menu-item action named “Query CMDB” will use a query definition file named “query\_cmdb”.

The query definition file contains

* An optional section ‘vars’ that can be used to calculate variables used in the expression; and which also supports special values:
  + “http-headers”: a dictionary of HTTP headers that should be set in the request.
  + “http-method”: one of “GET”, “POST”, “PUT” or “DELETE”.
  + “http-body”: a JSON body to send with a POST request.
* A section ‘query’ with
  + “expression”: the URL.
  + Optionally, “extract\_results\_from”: a value such as “result” if the JSON result is a list of values, and you want to update Resilient with each; or e.g. “result.content” if the JSON result contains {“content”: [ …a list of values ]}.
* One or more sections “incident\_fields”, “artifacts”, “tasks”, “notes”, “datatables”, or “attachment”, to define how Resilient is updated with the result.

Each of these values in the query definition file can use JINJA template syntax to substitute expressions from the incident into the query, and from the result into the Resilient update.

A few example files are provided showing various styles of query and update. These can be found in /usr/lib/python2.7/site-packages/query\_runner/data/queries\_rest/ (the exact path may vary depending on your Python installation).

After editing a query definition file, you don’t need to restart anything; these files are read from disk each time the query is executed.