



# Incident Response Platform Integrations

# ODBC Query Function V1.0.0

Release Date: May 2018

Resilient Functions simplify development of integrations by wrapping each activity into an individual workflow component. These components can be easily installed, then used and combined in Resilient workflows. The Resilient platform sends data to the function component that performs an activity then returns the results to the workflow. The results can be acted upon by scripts, rules, and workflow decision points to dynamically orchestrate the security incident response activities.

This guide describes the ODBC Query Function.

Overview

The ODBC Query Function establishes OBDC connection to the desired SQL database server and executes SELECT, INSERT, UPDATE or DELETE SQL statements.

This package includes a Script file that creates a test SQL table with dummy data, example workflows that demonstrate how to call these four SQL statements, rules that start the example workflows and a custom Resilient data table updated by the SELECT statement workflow.

The remainder of this document describes the included function, how to configure it in custom workflows, and any additional customization options.

Installation

Before installing, verify that your environment meets the following prerequisites:

* Resilient platform is version 30 or later.
* You have a Resilient account to use for the integrations. This can be any account that has the permission to view and modify administrator and customization settings, and read and update incidents. You need to know the account username and password.
* You have access to the command line of the Resilient appliance, which hosts the Resilient platform; or to a separate integration server where you will deploy and run the functions code. If using a separate integration server, you must install Python version 2.7.10 or later, or version 3.6 or later, and “pip”. (The Resilient appliance is preconfigured with a suitable version of Python.)

Install the Python components

The functions package contains Python components that will be called by the Resilient platform to execute the functions during your workflows. These components run in the ‘resilient-circuits’ integration framework.

The package also includes Resilient customizations that will be imported into the platform later.

Ensure that the environment is up to date,

sudo pip install --upgrade pip

sudo pip install --upgrade setuptools

sudo pip install --upgrade resilient-circuits

To install the package, you must first unzip it then install the package as follows:

sudo pip install --upgrade fn\_odbc\_query-<*version*>.tar.gz

Configure the Python components

The ‘resilient-circuits’ components run as an unprivileged user, typically named `integration`. If you do not already have an `integration` user configured on your appliance, create it now.

Perform the following to configure and run the integration:

1. Using sudo, become the integration user.

sudo su - integration

1. Use one of the following commands to create or update the resilient-circuits configuration file. Use –c for new environments or –u for existing environments.

resilient-circuits config -c

or

resilient-circuits config -u

1. Edit the resilient-circuits configuration file.
   1. In the [resilient] section, ensure that you provide all the information needed to connect to the Resilient platform.
   2. In the [fn\_odbc\_query] section, edit the settings as follows:

# Define your connection string

sql\_connection\_string=Driver={PostgreSQL};Server=IP Adress;Port=5432;

Database=myDataBase;Uid=myUserName;Pwd=myPassword;

# Optional setting

# Define a list of restricted SQL statements, separated by a comma.

# Example ["delete", "update", "insert"].

# Comment this line if there are no restrictions.

sql\_restricted\_sql\_statements=["delete", "insert", "update"]

# Optional setting

# Define if you wish to execute commits automatically after every SQL statement.

# Comment this line to use false - the default.

sql\_autocommit=true

# Optional setting

# Define a query timeout in seconds.

# Comment this line to use the default 0, which means "no timeout".

sql\_query\_timeout=10

# Optional setting

# Encoding and decoding settings needed for your SQL database.

# Choose one of the supported database types MariaDB, PostgreSQL or MySQL.

# Define which one of supported SQL Server database settings you want to use.

# At the moment MariaDB, PostgreSQL and MySQL are supported.

# Comment this line if you don't wish to configure decoding/encoding.

sql\_database\_type=MariaDB

# Optional setting

# Define number of rows to fetch.

# Comment this line to fetch all.

sql\_number\_of\_records\_returned=10

Connecting to a database

Connections to databases are made through the use of connection strings, which are driver-specific. General connection string information for most databases is available here [http://www.connectionstrings.com](http://www.connectionstrings.com/).

A standard PostgreSQL connection string:

Driver={PostgreSQL};Server=IP address;Port=5432;Database=myDataBase;  
Uid=myUsername;Pwd=myPassword;

Standard MySQL connection string:

Server=myServerAddress;Port=3306;Database=myDataBase;Uid=myUsername;  
Pwd=myPassword;

Deploy customizations to the Resilient platform

The package contains function definitions that you can use in workflows and includes example workflows and rules that show how to use these functions.

Deploy these customizations to the Resilient platform with the following command:

resilient-circuits customize

Answer the prompts to deploy functions, message destinations, workflows and rules.

Run the integration framework

To test the integration package before running it in a production environment, you must run the integration manually with the following command:

resilient-circuits run

The resilient-circuits command starts, loads its components, and continues to run until interrupted. If it stops immediately with an error message, check your configuration values and retry.

Configuration of resilient-circuits for restartability

For normal operation, resilient-circuits must run continuously. The recommend way to do this is to configure it to automatically run at startup. On a Red Hat appliance, this is done using a systemd unit file such as the one below. You may need to change the paths to your working directory and app.config.

The unit file should be named ‘resilient\_circuits.service’:

sudo vi /etc/systemd/system/resilient\_circuits.service

The contents:

[Unit]  
Description=Resilient-Circuits Service  
After=resilient.service  
Requires=resilient.service

[Service]  
Type=simple  
User=integration  
WorkingDirectory=/home/integration  
ExecStart=/usr/local/bin/resilient-circuits run  
Restart=always  
TimeoutSec=10  
Environment=APP\_CONFIG\_FILE=/home/integration/.resilient/app.config  
Environment=APP\_LOCK\_FILE=/home/integration/.resilient/resilient\_circuits.lock

[Install]  
WantedBy=multi-user.target

Ensure that the service unit file is correctly permissioned:

sudo chmod 664 /etc/systemd/system/resilient\_circuits.service

Use the systemctl command to manually start, stop, restart and return status on the service:

sudo systemctl resilient\_circuits [start|stop|restart|status]

Log files for systemd and the resilient-circuits service can be viewed through the journalctl command:

sudo journalctl -u resilient\_circuits --since "2 hours ago"

Function Descriptions

Once the function package deploys the function, you can view it in the Resilient platform Functions tab, as shown below. The package also includes example workflows and rules that show how the functions can be used. You can copy and modify these workflows and rules for your own needs.

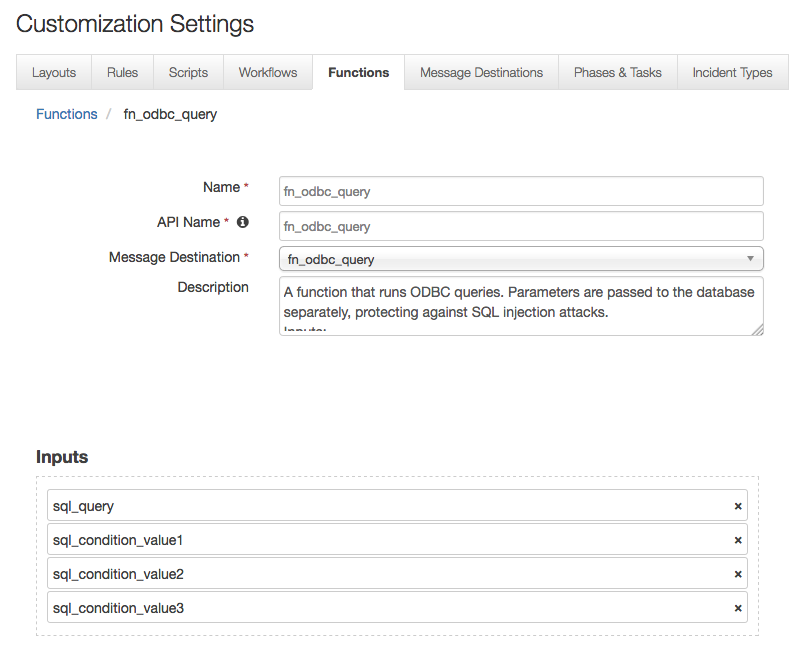


Figure 1: ODBC Function

ODBC Query: fn\_odbc\_query

A function that runs ODBC queries. Query and the data/parameters are passed to the database separately, protecting against SQL injection attacks.

As shown above, this function takes the following input fields:

* sql\_query: Query to perform. It is a “Text with value String” type of filed and it contains four predefined example SQL statements that you can choose and also edit from within the workflow. The predefined queries, as shown in Figure 2, contain parameters using a question mark as a place holder in the SQL query. Users can set values for the question marks by setting them in input fields sql\_condition\_value1, sql\_condition\_value2 and sql\_condition\_value3 in the workflow.
* sql\_condition\_value1, sql\_condition\_value2 and sql\_condition\_value3: Parameters used in the query.

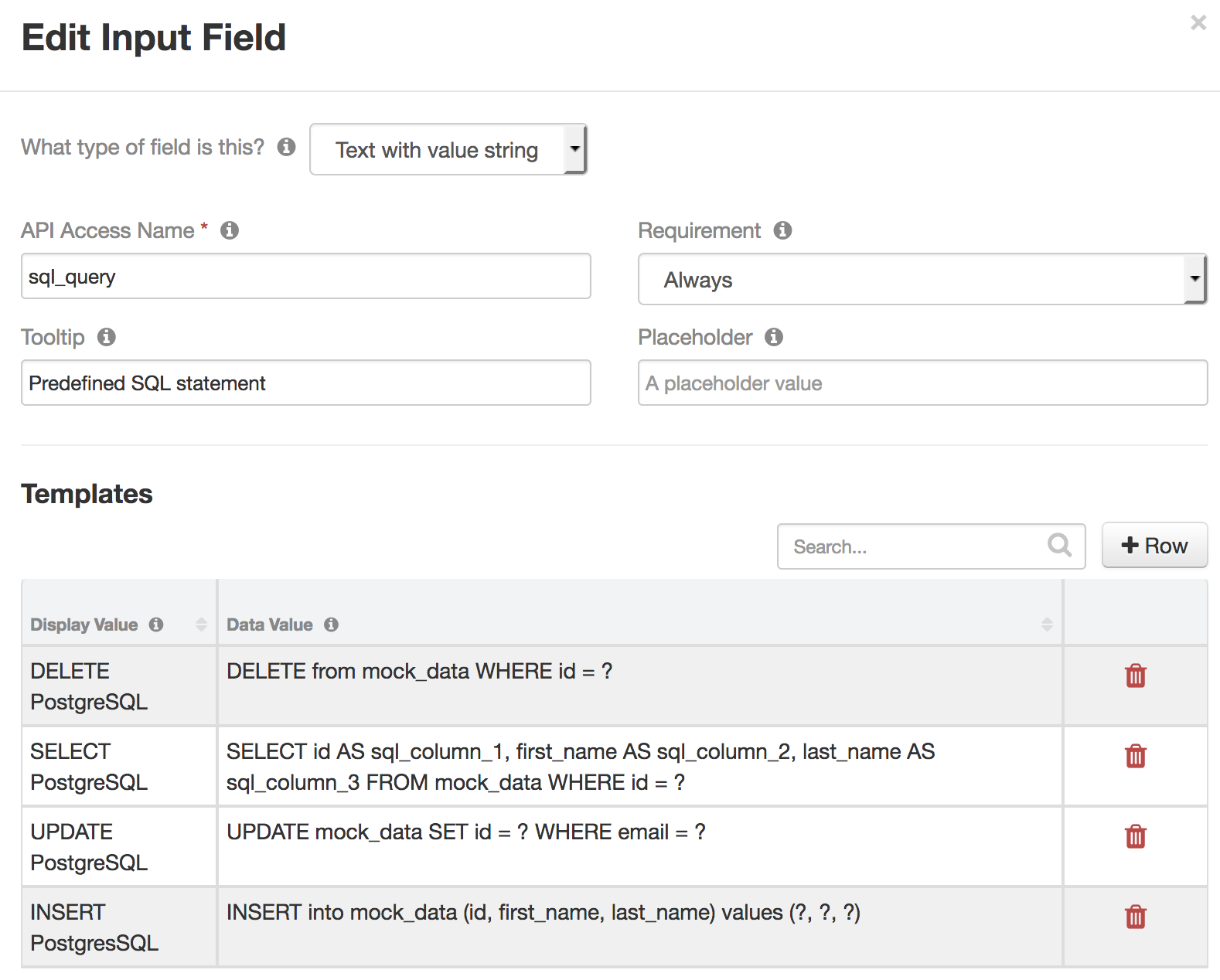


Figure 2: Predefined SQL statements for the 'sql\_query' input field

Example ODBC SELECT PostgreSQL

The example workflow (object type = Artifact) that calls this function is “Example ODBC SELECT PostgreSQL”. The Input tab of this function is shown in Figure 3. When defining the SQL table column names that SELECT query will return it’s important to use alias column syntax. Alias names need to match Resilient data table field names. This will make sure query results are saved in the proper Resilient data table field.

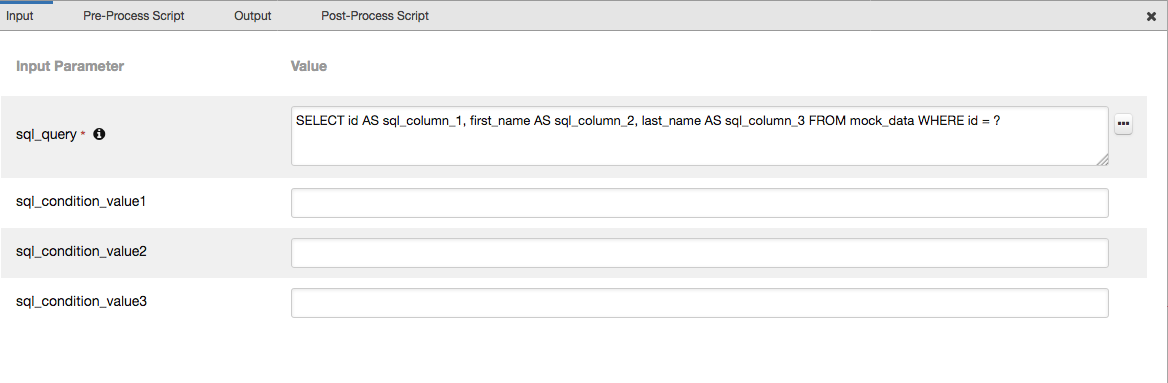


Figure 3: Example SQL SELECT statement

Users may insert data using parameters sql\_condition\_value1, sql\_condition\_value2 and sql\_condition\_value3 on the Input tab or set them in the Pre-Process Script to the value of the artifact associated with this workflow as shown in Figure 4.

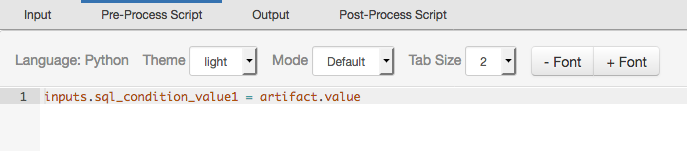


Figure 4: Example Pre-process Script for SELECT, DELETE, INSERT statements

A Menu Item rule called “Example ODBC SELECT PostgreSQL” is also included. This rule calls the provided workflow.

With these components in place, users can click the Actions button and the above rule appears. By clicking the menu item, the ODBC Query function is activated. The query results update the custom Resilient data table called “sql\_query\_results\_dt” shown in Figure 5. The definition of this data table is also included in the package.

Fist two data table fields are set automatically, ‘sql\_artifact\_value’ is set to the value of the artifact associated with this workflow and ‘sql\_timestamp’ is set to date and time of the function call. All other data table fields ‘sql\_column\_1’ to ‘sql\_column\_5’ are updated by the query results.

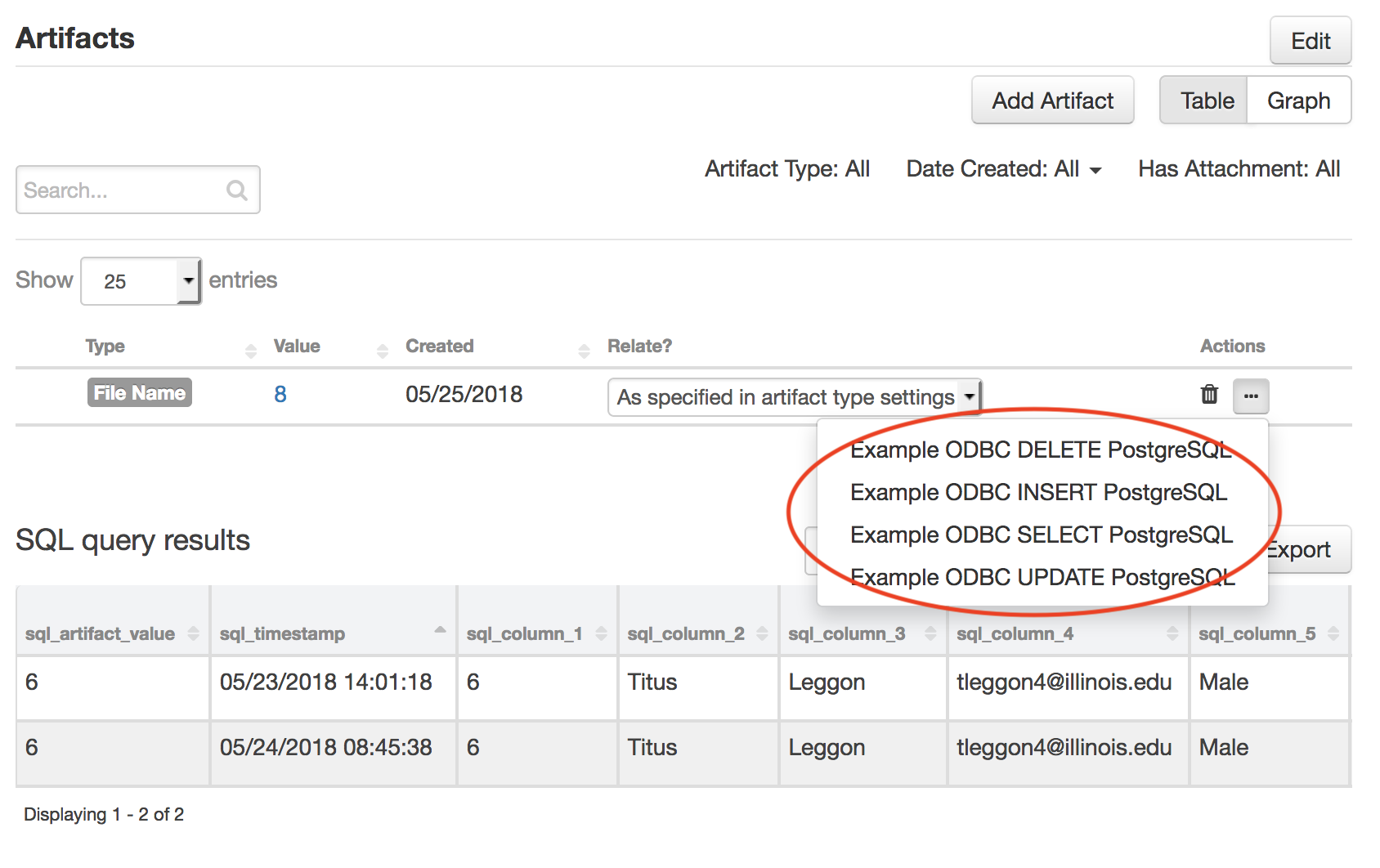


Figure 5: Menu items and Data table

Users may modify the “sql\_query\_results\_dt” data table, add or remove any fields. To ensure SQL query results are saved in proper Resilient data table fields, users need to modify the RESILENT\_DATATABLE\_COLUMN\_NAMES\_LIST in the Post-Process Script as shown in Figure 6.

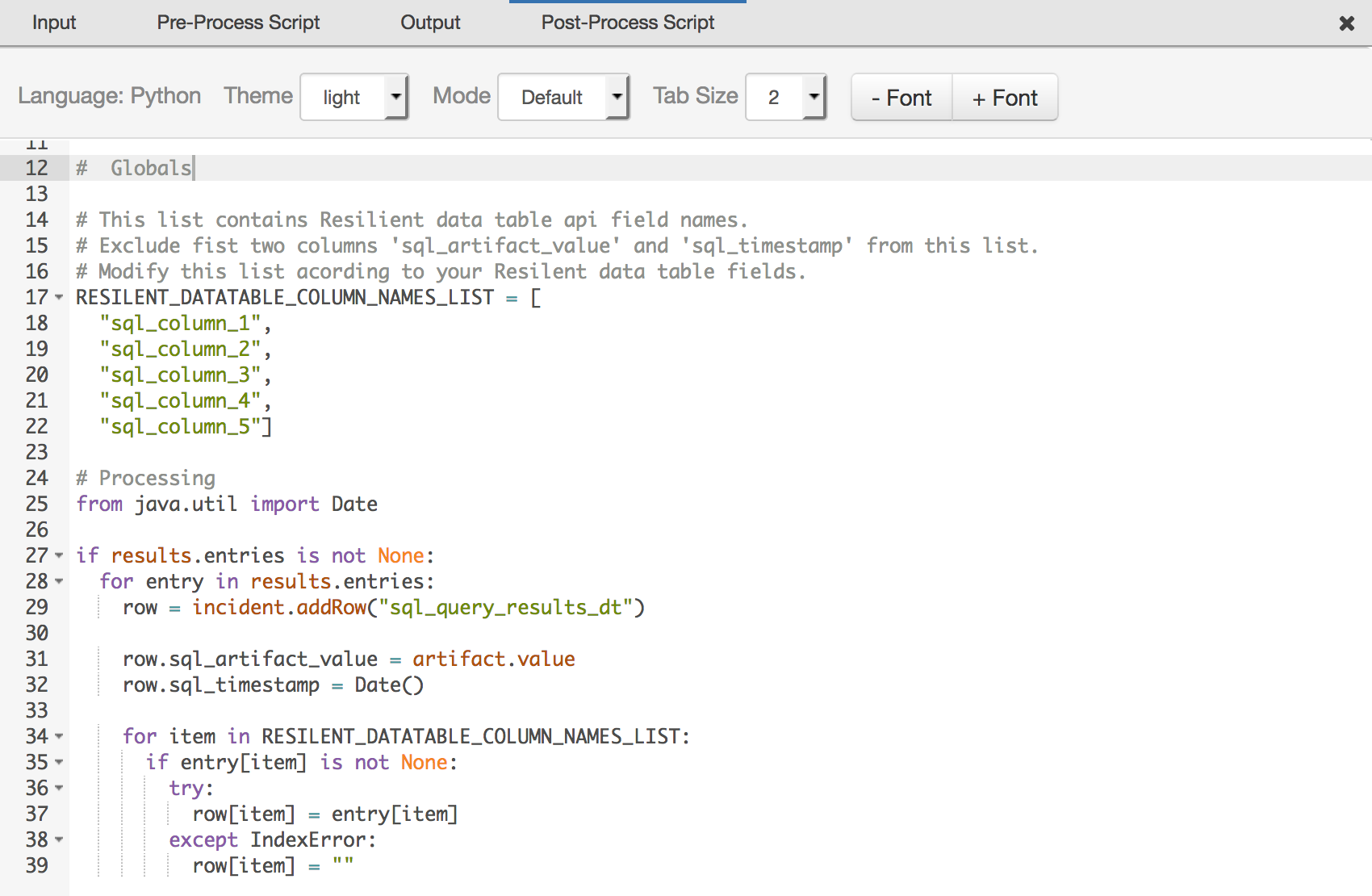


Figure 6: Post-Process Script for SELECT statement

Example ODBC UPDATE PostgreSQL

The example workflow (object type = Artifact) that calls this function is “Example ODBC UPDATE PostgreSQL”. The Input tab of this function is shown in Figure 6.

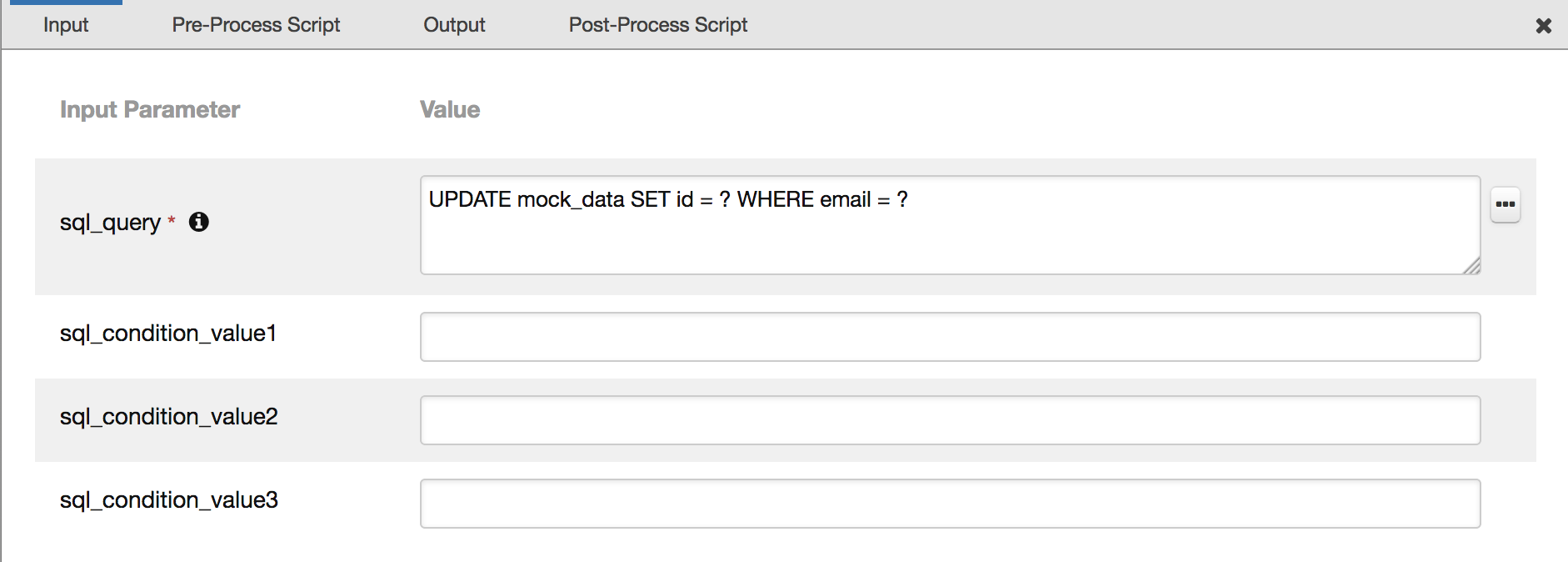


Figure 7: Example of SQL UPDATE statement

Users may insert data using parameters sql\_condition\_value1, sql\_condition\_value2 and sql\_condition\_value3 on the Input tab or set them in the Pre-Process Script to the value and description of the artifact associated with this workflow as shown in Figure 7.

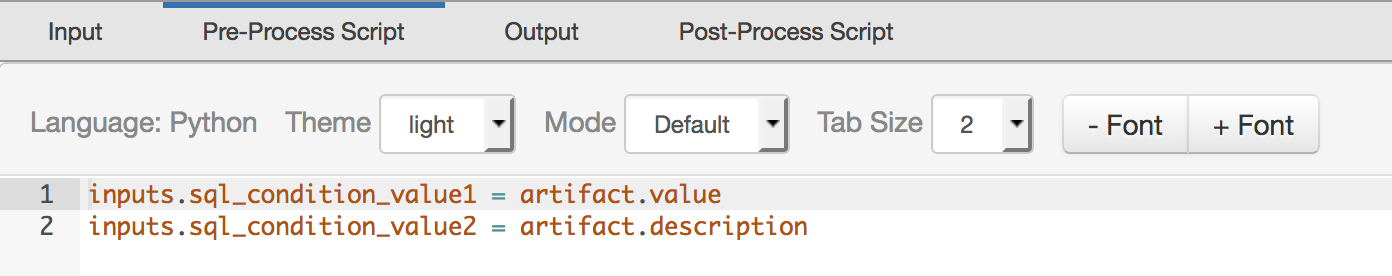


Figure 8: Pre-process Script for UPDATE statement

A Menu Item rule called “Example ODBC UPDATE PostgreSQL” is also included. This rule calls the provided workflow.

Users can click the Actions button and the above rule appears as shown in Figure 5. By clicking the menu item, the ODBC Query function is activated. The function will return number of processed rows and display the message in the Action Status.

Example ODBC DELETE PostgreSQL

The example workflow (object type = Artifact) that calls this function is “Example ODBC DELETE PostgreSQL”. The Input tab of this function is shown in Figure 8.

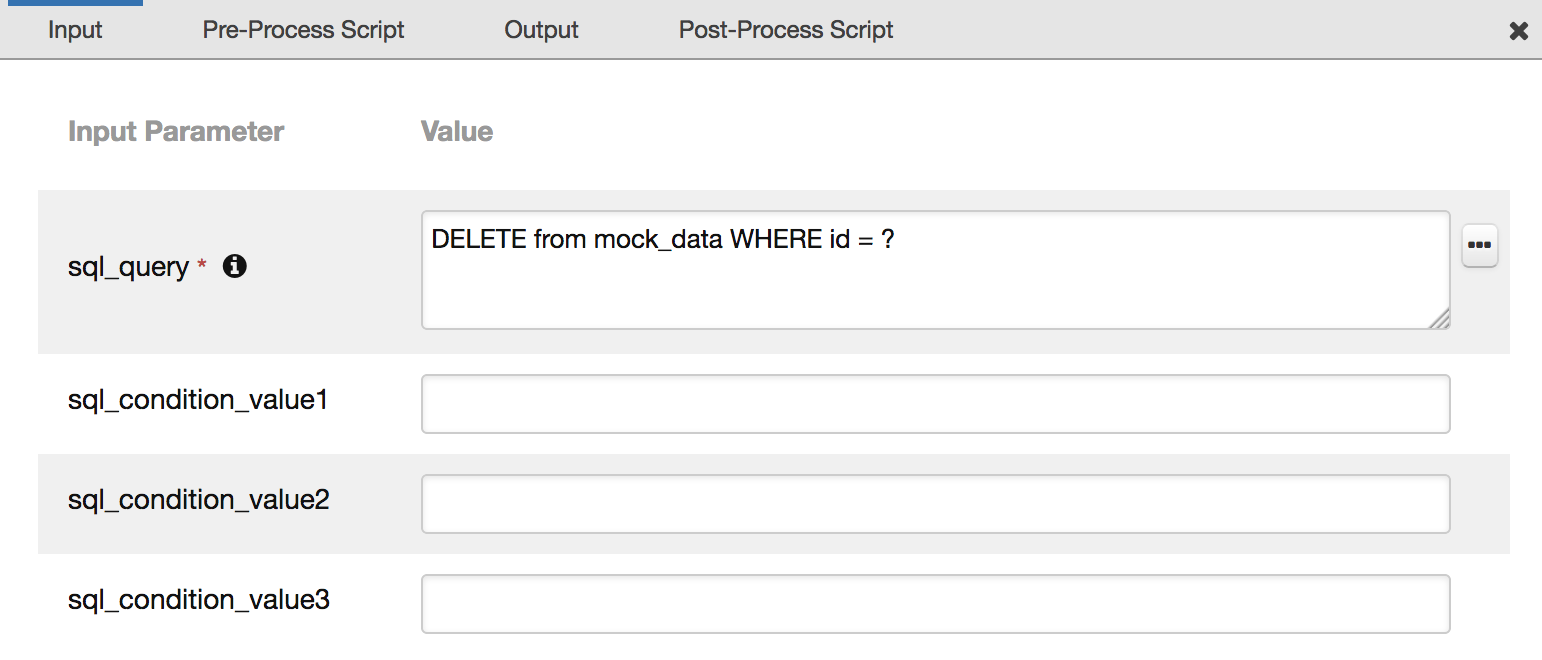


Figure 9: Example of SQL DELETE statement

Users may insert data using parameter sql\_condition\_value1 on the Input tab or set it in the Pre-Process Script to the value of the artifact associated with this workflow as shown in Figure 4.

A Menu Item rule called “Example ODBC DELETE PostgreSQL” is also included. This rule calls the provided workflow.

Users can click the Actions button and the above rule appears as shown in Figure 5. By clicking the menu item, the ODBC Query function is activated. The function will return number of processed rows and display the message in the Action Status.

Example ODBC INSERT PostgreSQL

The example workflow (object type = Artifact) that calls this function is “Example ODBC INSERT PostgreSQL”. The Input tab of this function is shown in Figure 9.

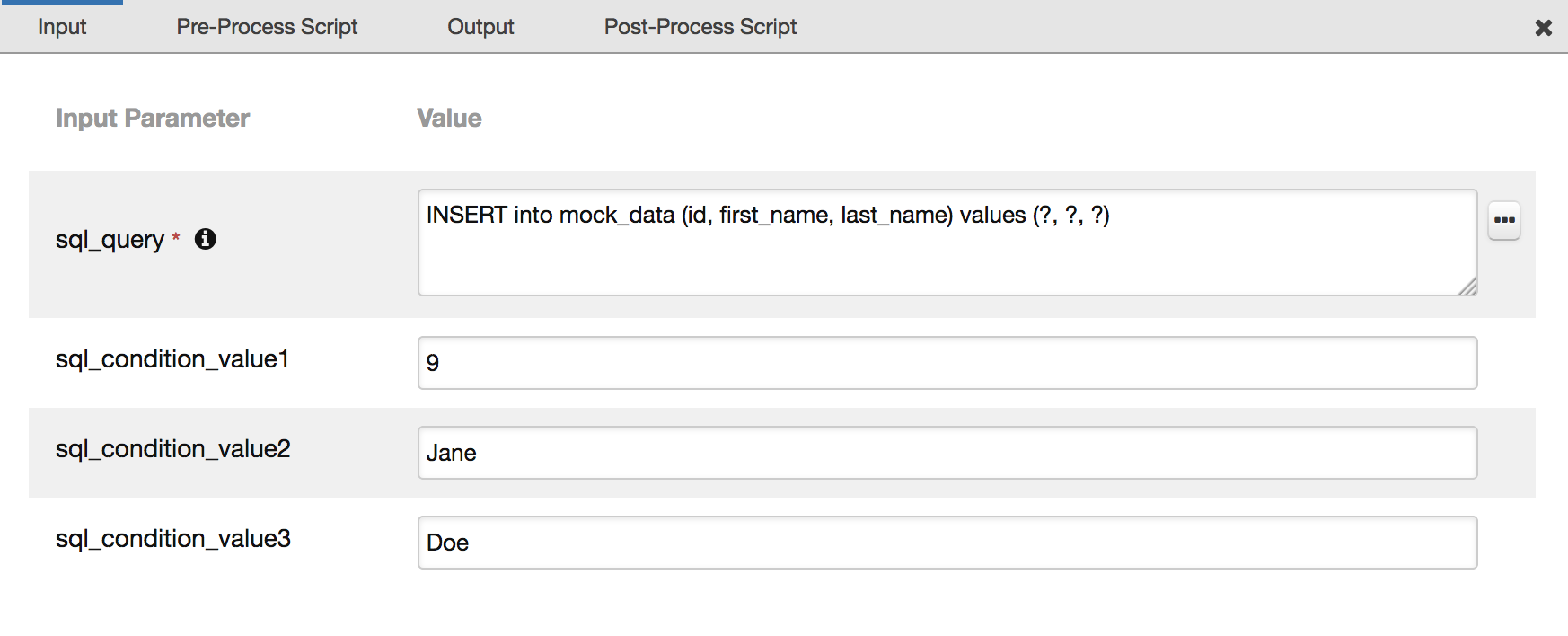


Figure 10: Example of SQL INSERT statement

Users may insert data using parameters sql\_condition\_value1, sql\_condition\_value2 and sql\_condition\_value3 on the Input tab as shown in Figure 9 or set them in the Pre-Process Script to the value of the artifact associated with this workflow as shown in Figure 3.

A Menu Item rule called “Example ODBC INSERT PostgreSQL” is also included. This rule calls the provided workflow.

Users can click the Actions button and the above rule appears as shown in Figure 5. By clicking the menu item, the ODBC Query function is activated. The function will return number of processed rows and display the message in the Action Status.

Resilient Platform Configuration

<Only use this section if you have general configuration advice or requirements, such as describing issues moving the configuration from a testing environment to a production environment. Think about the parameters that need to be modified per each customer’s environment.>

<If your function exports data tables or custom fields, the changes in the Resilient platform layouts may not be preserved. Therefore, you should document those changes in this section and advise the reader to manually make those changes in the platform.>

<If there are scenarios of use which would help a customer understand how to use the Integration as a whole (not per function), indicate them here.>

<If not needed, delete this section.>

Troubleshooting

There are several ways to verify the successful operation of a function.

* Resilient Action Status

When viewing an incident, use the Actions menu to view Action Status. By default, pending and errors are displayed. Modify the filter for actions to also show Completed actions. Clicking on an action displays additional information on the progress made or what error occurred.

* Resilient Scripting Log

A separate log file is available to review scripting errors. This is useful when issues occur in the pre-processing or post-processing scripts. The default location for this log file is: /var/log/resilient-scripting/resilient-scripting.log.

* Resilient Logs

By default, Resilient logs are retained at /usr/share/co3/logs. The client.log may contain additional information regarding the execution of functions.

* Resilient-Circuits

The log is controlled in the .resilient/app.config file under the section [resilient] and the property logdir. The default file name is app.log. Each function will create progress information. Failures will show up as errors and may contain python trace statements.

Support

For additional support, contact [support@resilientsystems.com](mailto:support@resilientsystems.com).

Including relevant information from the log files will help us resolve your issue.

*Documentation Guidelines*

*<Do NOT include this section in your guide.>*

*Here are some writing guidelines:*

* *Never use “Resilient,” instead use “Resilient platform.”*
* *Use “deploy to the Resilient platform” to describe the resilient-circuits customize command.*
* *Do not initial cap function, workflow, etc. unless you are referring to a specific item (proper name), such as Utilities Function.*
* *Try to avoid passive voice and future tense.*
* *For the guide’s file name, use this format: Resilient Integration <name> Function*
* *In the Word file, open properties and make these changes:*
  + *Owner = IBM Resilient*
  + *Title = Resilient IRP Integrations <name> Function Guide*

*If you don’t know how to open properties:*

1. *Click* ***File > Info****.*
2. *On the right side of the Info page, click the* ***Properties*** *drop-down and select* ***Show Document Panel****. This shows the Document Properties with the Author and Title fields.*