

User Guide: fn_datatable_utils_v1.1.0

Table of Contents

- [Setup](#)
- [Key Features](#)
- [Function - Data Table Utils: Get Row](#)
- [Function - Data Table Utils: Get Rows](#)
- [Function - Data Table Utils: Update Row](#)
- [Function - Data Table Utils: Delete Row](#)
- [Function - Data Table Utils: Delete Rows](#)
- [Function - Data Table Utils: Create CSV Datatable](#)
- [Data Table - Example CSV Datatable](#)
- [Rules](#)

Release Notes

v1.1.0

- Added support for App Host Added Functions:
- `dt_utils_get_row`
- `dt_utils_get_rows`
- `dt_utils_delete_row`
- `dt_utils_delete_rows`
- `dt_utils_create_csv_table`

v1.0.0

- Initial Release

This package contains 6 functions that help you manipulate IBM Resilient Data Tables

Functions

New Function

data table util

Name	Description	
Data Table Utils: Create CSV Datatable	Add CVS data to a named datatable.	
Data Table Utils: Delete Row	Function that deletes a row from a Data Table given the internal row ID.	
Data Table Utils: Delete Rows	Function that deletes rows from a Data Table given a list of internal row IDs or a 'search_column and search_value' pair.	
Data Table Utils: Get Row	Function that searches for a row using a internal row ID or a 'search_column and search_value' pair and returns the cells of the row that is found, if such a row exists.	
Data Table Utils: Get Rows	Function that returns the full unsorted list of JSON objects which contain all information regarding each row found, if no searching/sorting criteria were provided.	
Data Table Utils: Update Row	Function that takes a JSON String of 'search_column and search_value' pairs to update a Data Table row.	

The 6 functions allow you to GET, UPDATE, DELETE a row, GET and DELETE rows in a Data Table, and populate CSV data into a datatable.

Setup

To reference the example datatable, create a new incident tab and drag the **Example CSV DataTable** into the widget area.

The screenshot shows the 'Customization Settings' page in the Resilient interface. The 'Layouts' tab is selected, and the 'Incident Tabs' section is expanded. The 'Manage Tabs' tab is active, showing a list of incident tabs: Tasks, Details, Breach, Notes, Mem..., News..., Attac..., Stats, Timeline, and Artifacts. The 'Tasks' tab is selected, and its configuration is shown: Tab Text is 'Tasks', and Tab Visible is set to 'Yes'. A red box highlights the 'Add Tab' button in the left sidebar. Below the screenshot, a diagram shows the 'Incident: csv datatable' configuration. The 'Example CSV Datatable' is selected in the 'Data Tables' list, and a red arrow points to it from the 'Example CSV Datatable' entry in the 'Data Tables' list.

Resilient Dashboards Simulations Incidents **Create** Admin User ResOrg

Customization Settings

Layouts Rules Scripts Workflows Functions Message Destinations Phases & Tasks Incident Types Breach Artifacts

New Incident Wizard

Incident Tabs

- Manage Tabs
- Summary Section
- Tasks
- Details
- Breach
- Notes
- Members
- News Feed
- Attachments
- Stats
- Timeline
- Artifacts
- Email
- + Add Tab**

Close Incident

Incident: **Manage Tabs** Cancel Save

Tasks Details Breach Notes Mem... News... Attac... Stats Timeline Artifacts

Email +

Tab Text * Tasks

Tab Visible ☒ Yes ☐ No ☐ Conditional

Layouts Rules Scripts Workflows Functions Message Destinations Phases & Tasks Incident Types Breach Artifacts

Incident: csv datatable Delete Save

Example CSV Datatable

Fields

- Address
- Alberta Health Risk Assessment
- Assessed Liability
- City
- Country/Region
- Created By
- Criminal Activity
- Customizations Field (internal)
- Data Encrypted

Add Field

Data Tables

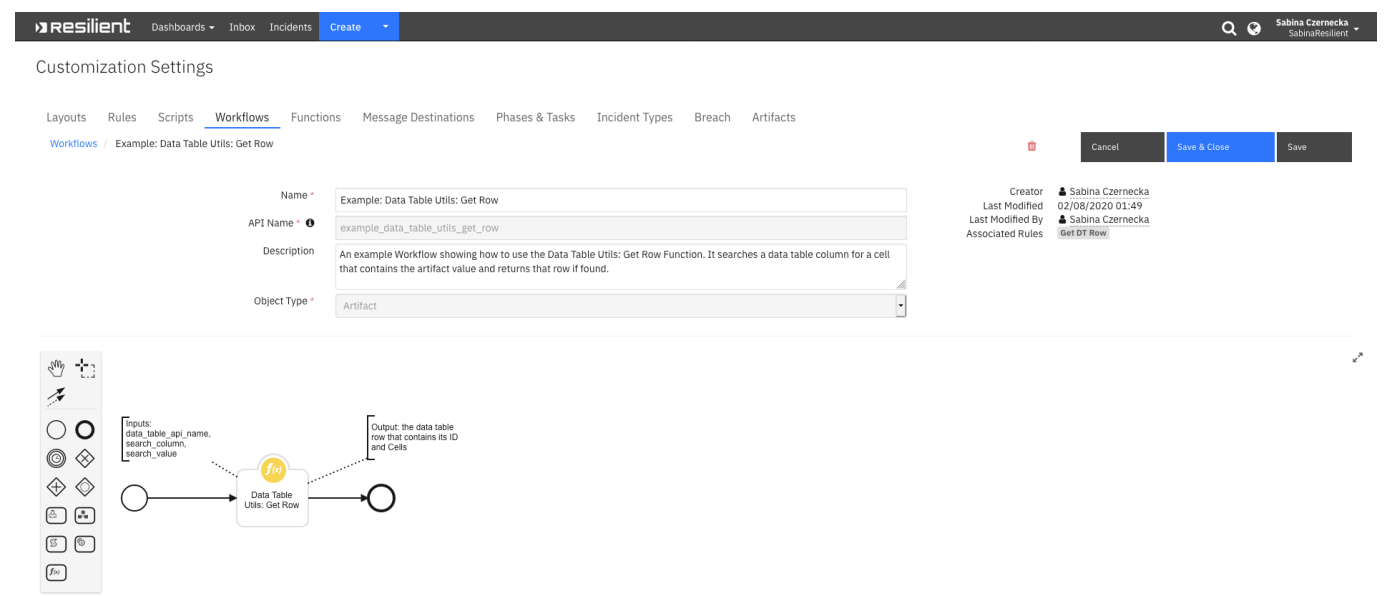
- Carbon Black Query Results
- Example CSV Datatable**

Add Table

Function - Data Table Utils: Get Row

Function that searches for a row using a internal row ID or a search_column and search_value pair, and returns the information on the row that is found, if such a row exists.

An example Rule and Workflow exist for using this function on the example datatable from an artifact value.



► Inputs:

Name	Type	Required	Example	Tooltip
dt_utils_datatable_api_name	text	Yes	—	The API name of the Data Table
dt_utils_row_id	number	No	—	The internal ID of the row to be retrieved
dt_utils_search_column	text	No	—	The API name of the column to search
dt_utils_search_value	text	No	—	The cell value to search for within the search column
incident_id	number	Yes	—	—

► Outputs:

```
results = {
  # TODO: Copy and paste an example of the Function Output within this
  # code block.
  # To view the output of a Function, run resilient-circuits in DEBUG
  # mode and invoke the Function.
  # The Function results will be printed in the logs: "resilient-
  # circuits run --loglevel=DEBUG"
}
```

► Workflows

► Example Pre-Process Script:

```
# Data Table Utils: Example: Get Row

#####
### Define Inputs ###
#####

# The ID of this incident
inputs.incident_id = incident.id

# The api name of the Data Table to update
inputs.dt_utils_datatable_api_name = "dt_utils_test_data_table"

# The column api name to search for
inputs.dt_utils_search_column = "dt_col_name"

# The cell value to search for
inputs.dt_utils_search_value = artifact.value

## Alternatively you can get the row by its ID by defining this input:
# inputs.dt_utils_row_id = 3
```

► Example Post-Process Script:

```
search_value = results.inputs["dt_utils_search_value"]
note_text = u"<b>Result from Example: Data Table Utils: Get Row</b><br>
search value: {0}".format(search_value)
if results.success:
note_text = u"{0} <br>{1}".format(note_text, str(results["row"]))
else:
note_text = u"{0} <br>No row found.".format(note_text)

incident.addNote(helper.createRichText(note_text))
```

Function - Data Table Utils: Get Rows

Function that returns the full list of rows in a datatable based on the search value. List sorting is possible using the sort_by and sort_direction input fields.

An example Rule and Workflow exist for searching the example datatable based on an artifact value.

resilient

Dashboards

Inbox

Incidents

Create

Search

Sabina Czernecka

Sabina@resilient

Customization Settings

Layouts

Rules

Scripts

Workflows

Functions

Message Destinations

Phases & Tasks

Incident Types

Breach

Artifacts

Workflows / Example: Data Table Utils: Get Rows

Name *

Example: Data Table Utils: Get Rows

API Name *

example_data_table_utils_get_rows

Description

An example Workflow showing how to use the Data Table Utils: Get Rows Function.

Object Type *

Artifact

Cancel

Save & Close

Save

Creator

Sabina Czernecka

Last Modified

01/08/2020 23:06

Last Modified By

Sabina Czernecka

Associated Rules

Get DT Rows

Inputs:

datatable_api_name,
max_rows, sort_by,
sort_direction,
search_column,
search_value

Output: The
Data Table rows

Workflow Diagram

Graph showing a function node 'Data Table Utils: Get Rows' with inputs and outputs.

► Inputs:

Name	Type	Required	Example	Tooltip
dt_utils_datatable_api_name	text	Yes	—	The API name of the Data Table
dt_utils_max_rows	number	No	—	The maximum number of rows to be returned
dt_utils_search_column	text	No	—	The API name of the column to search
dt_utils_search_value	text	No	—	The cell value to search for within the search column
dt_utils_sort_by	text	No	—	The API name of the column to sort by
dt_utils_sort_direction	select	No	—	—
incident_id	number	Yes	—	—

► Outputs:

```
results = {  
    # TODO: Copy and paste an example of the Function Output within this  
    # code block.  
    # To view the output of a Function, run resilient-circuits in DEBUG  
    # mode and invoke the Function.  
    # The Function results will be printed in the logs: "resilient-  
    # circuits run --loglevel=DEBUG"  
}
```

► Workflows

► Example Pre-Process Script:

```
# Data Table Utils: Example: Get Rows

#####
### Define Inputs ###
#####

# The ID of this incident
inputs.incident_id = incident.id

# The api name of the Data Table to update
inputs.dt_utils_datatable_api_name = "dt_utils_test_data_table"

# The number of max rows to return
if rule.properties.dt_utils_max_rows:
    inputs.dt_utils_max_rows = rule.properties.dt_utils_max_rows
else:
    inputs.dt_utils_max_rows = 0

# The direction of the sort
if rule.properties.dt_utils_sort_direction:
    inputs.dt_utils_sort_direction = rule.properties.dt_utils_sort_direction
else:
    inputs.dt_utils_sort_direction = "ASC"

# The api name of the column to sort by
if rule.properties.dt_utils_sort_by:
    inputs.dt_utils_sort_by = rule.properties.dt_utils_sort_by
else:
    inputs.dt_utils_sort_by = None

# The column api name to search for
inputs.dt_utils_search_column = "dt_col_name"

# The cell value to search for
inputs.dt_utils_search_value = artifact.value
```

► Example Post-Process Script:

```
if not results.success:
    incident.addNote(helper.createRichText("<b>Result from Example: Data Table  
Utils: Delete Rows</b><br>No rows found."))
```

Function - Data Table Utils: Update Row

Function that takes a string-encoded JSON String of 'search_column and search_value' pairs to update a Data Table row.

When used on a datatable, specify `dt_utils_row_id = 0` to refer to the currently referenced datatable row.

Two sets example Rule and Workflow are available for changing the example datatable from an artifact value and directly from a row in the datatable.

resilient

DashboardsInboxIncidentsCreate

Sabina CzerneckaSabina@resilient

Customization Settings

LayoutsRulesScriptsWorkflowsFunctionsMessage DestinationsPhases & TasksIncident TypesBreachArtifacts

WorkflowsExample: Data Table Utils: Update Row

NameExample: Data Table Utils: Update Row

API Nameexample_data_table_utils_update_row

DescriptionAn example Workflow showing how to use the Data Table Utils: Update Row Function. It uses an Artifact value to search and find a row in the data table. Then updates the cells with the given values.

Object TypeArtifact

CreatorSabina Czernecka

Last Modified15/07/2020 12:11

Last Modified ByAPI_ALL

Associated RulesUpdate DT Row

Inputs: data, table, api_name, search_column, search_value

Output: the data table row that contains its ID and Cells

Data Table Utils: Get Row

✕

Data Table Utils: Update Row

○

If we find a row, continue, else end

Inputs: data, table, api_name, row_id, cells_to_update

Output: the updated row

► Inputs:

Name	Type	Required	Example	Tooltip
dt_utils_cells_to_update	text	Yes	—	A JSON String containing the column names and cell values to update
dt_utils_datatable_api_name	text	Yes	—	The API name of the Data Table
dt_utils_row_id	number	No	—	The internal ID of the row to be retrieved
incident_id	number	Yes	—	—

► Outputs:

```
results = {
  # TODO: Copy and paste an example of the Function Output within this
  # code block.
  # To view the output of a Function, run resilient-circuits in DEBUG
  # mode and invoke the Function.
  # The Function results will be printed in the logs: "resilient-
  # circuits run --loglevel=DEBUG"
}
```

- Workflows
- Example Pre-Process Script:

7 / 18

```
# Data Table Utils: Example: Update Row
import java.util.Date as Date

#####
### Define pre-processing functions ###
#####

def dict_to_json_str(d):
    """Function that converts a dictionary into a JSON string.
    Supports types: basestring, bool, int and nested dicts.
    Does not support lists.
    If the value is None, it sets it to False."""

    json_entry = '{"0}":{1}'
    json_entry_str = '{"0}":"' + '{1}' + '"'
    entries = []

    for entry in d:
        key = entry
        value = d[entry]

        if value is None:
            value = False

        if isinstance(value, list):
            helper.fail('dict_to_json_str does not support Python Lists')

        if isinstance(value, basestring):
            value = value.replace(u'"', u'\\\"')
            entries.append(json_entry_str.format(key, value))

        elif isinstance(value, bool):
            value = 'true' if value == True else 'false'
            entries.append(json_entry.format(key, value))

        elif isinstance(value, dict):
            entries.append(json_entry.format(key, dict_to_json_str(value)))

        else:
            entries.append(json_entry.format(key, value))

    return '{0} {1} {2}'.format('{', ','.join(entries), '}')

# S T A R T

# The ID of this incident
inputs.incident_id = incident.id

# The api name of the Data Table to update [here it is taken from previous
Get Row Function]
inputs.dt_utils_datatable_api_name = "dt_utils_test_data_table"

# Refer to the existing row (value: 0)
```



```
inputs.dt_utils_row_id = 0

# The column api names and the value to update the cell to
inputs.dt_utils_cells_to_update = dict_to_json_str({
  "datetime": Date().getTime(),
  "text": "Done"
})
```

► Example Post-Process Script:

None

Function - Data Table Utils: Delete Row

Function that deletes a row from a Data Table given the internal row ID.

When used on a datatable, specify dt_utils_row_id = 0 to reference the currently referenced datatable row. The delete operation will be delayed as the workflow will first terminate before the row is deleted.

An example Rule and Workflow are available for deleting datatable rows based on an artifact value and against a row in the example datatable.

Resilient

Dashboard ▾InboxIncidentsCreate ▾

SearchSabina CzerneckaSabina@resilient

Customization Settings

LayoutsRulesScriptsWorkflowsFunctionsMessage DestinationsPhases & TasksIncident TypesBreachArtifacts

Workflows / Example: Data Table Utils: Delete Row

NameExample: Data Table Utils: Delete Row

API Nameexample_data_table_utils_delete_row

DescriptionAn example Workflow showing how to use the Data Table Utils: Delete Row Function. It uses an Artifact value to search and find a row in the data table. Then deletes that row.

Object TypeArtifact

CreatorSabina Czernecka

Last Modified15/07/2020 12:11

Last Modified ByAPI_ALL

Associated RulesDelete DT Row

Cancel

Save & Close

Save

Inputs: data_table_api_name, search_column, search_value

Output: the data table row that contains its ID and Cells

Inputs: data_table_api_name, row_id

Output: a success flag if row deleted

Start

Data Table Utils: Get Row

Decision

Data Table Utils: Delete Row

End

If we find a row, continue, else end

► Inputs:

Name	Type	Required	Example	Tooltip
dt_utils_datatable_api_name	text	Yes	–	The API name of the Data Table

Name	Type	Required	Example	Tooltip
<code>dt_utils_row_id</code>	number	No	–	The internal ID of the row to be retrieved. Specify 0 when used on a datatable.
<code>incident_id</code>	number	Yes	–	–

► Outputs:

```
results = {
    # TODO: Copy and paste an example of the Function Output within this
    # code block.
    # To view the output of a Function, run resilient-circuits in DEBUG
    # mode and invoke the Function.
    # The Function results will be printed in the logs: "resilient-
    # circuits run --loglevel=DEBUG"
}
```

► Workflows

► Example Pre-Process Script:

```
# Data Table Utils: Example: Delete Row

#####
### Define Inputs ###
#####

# The ID of this incident
inputs.incident_id = incident.id

# The api name of the Data Table [here it is taken from previous Get Row
# Function]
inputs.dt_utils_datatable_api_name =
workflow.properties.row_to_delete.inputs.dt_utils_datatable_api_name

# The ID of the row to delete [again, taken from previous Get Row
# Function]
inputs.dt_utils_row_id = workflow.properties.row_to_delete.row["id"]
```

► Example Post-Process Script:

```
# {'success': True, 'inputs': {'incident_id': 2150,
'dt_utils_datatable_api_name': 'dt_utils_test_data_table',
'dt_utils_row_id': 821}, 'row': {'success': True, 'title': None,
'message': None, 'hints': []}}
if results.success:
    note = u"Row id: {} removed from datatable: {} for artifact:
```

```

{}".format(results.inputs['dt_utils_row_id'],
results.inputs['dt_utils_datatable_api_name'], artifact.value)
else:
note = u"Artifact: {} not found in datatable: {}".format(artifact.value,
results.inputs['dt_utils_datatable_api_name'])

incident.addNote(note)

```

Function - Data Table Utils: Delete Rows

Function that deletes rows from a Data Table given a list of internal row IDs or a 'search_column and search_value' pair.

An example Rule and Workflow are available for deleting datatable rows based on an artifact value.

Customization Settings

Name: Example: Data Table Utils: Delete Rows

API Name: example_data_table_utils_delete_rows

Description: An example Workflow showing how to use the Data Table Utils: Delete Rows Function. It allows the user to delete rows using either internal row IDs or 'search_column and search_value' pair in a Data Table.

Object Type: Artifact

Creator: Sabina Czernecka
Last Modified: 01/08/2020 23:09
Last Modified By: Sabina Czernecka
Associated Rules: Delete DT Rows

Inputs: datatable_api_name, max_rows, sort_by, sort_direction, search_column, search_value

Output: The Data Table rows

Workflow Steps:

- Data Table Utils: Get Rows
- Decision: If we find a row, continue, else end
- Data Table Utils: Delete Rows
- Output: The list of internal row IDs of the Data Table that were deleted

► Inputs:

Name	Type	Required	Example	Tooltip
dt_utils_datatable_api_name	text	Yes	—	The API name of the Data Table
dt_utils_rows_ids	text	No	—	The list of internal rows IDs of a Data Table to delete
dt_utils_search_column	text	No	—	The API name of the column to search
dt_utils_search_value	text	No	—	The cell value to search for within the search column
incident_id	number	Yes	—	—

► Outputs:

```

results = {
    # TODO: Copy and paste an example of the Function Output within this
    # code block.
    # To view the output of a Function, run resilient-circuits in DEBUG
    # mode and invoke the Function.
    # The Function results will be printed in the logs: "resilient-
    # circuits run --loglevel=DEBUG"
}

```

► Workflows

► Example Pre-Process Script:

```

# Data Table Utils: Example: Delete Row

#####
### Define Inputs ###
#####

# The ID of this incident
inputs.incident_id = incident.id

# The api name of the Data Table, search column, search value [here it is
# taken from previous Get Rows Function inputs]
inputs.dt_utils_datatable_api_name =
workflow.properties.rows_to_delete.inputs.dt_utils_datatable_api_name

# The internal IDs of the rows that will be deleted [again, taken from
# previous Get Rows Function]
if workflow.properties.rows_to_delete and
workflow.properties.rows_to_delete.rows:
    rows_ids = []
    for row in workflow.properties.rows_to_delete.rows:
        rows_ids.append(row["id"])
    inputs.dt_utils_rows_ids = str(rows_ids)

```

► Example Post-Process Script:

```

if results.success:
    note = u"<b>Result from Example: Data Table Utils: Artifact: {} Delete
    Rows</b><br> {}".format(artifact.value, str(results["rows_ids"]))
else:
    note = u"<b>Result from Example: Data Table Utils: Artifact: {} not found
    in datatable: {}".format(artifact.value,
    results.inputs['dt_utils_datatable_api_name'])

incident.addNote(helper.createRichText(note))

####

```

```
# {'success': True, 'inputs': {'incident_id': 2150,
'dt_utils_datatable_api_name': 'dt_utils_test_data_table',
'dt_utils_row_id': 821}, 'row': {'success': True, 'title': None,
'message': None, 'hints': []}}
if results.success:
note = u"Row id: {} removed from datatable: {} for artifact:
{}".format(results.inputs['dt_utils_row_id'],
results.inputs['dt_utils_datatable_api_name'], artifact.value)
"""
```

Function - Data Table Utils: Create CSV Datatable

Add CVS data to a named datatable. CSV data can originate from another function or from a referenced attachment with CSV encoded data.

A mapping table is used to map CSV header row labels to datatable column (API) names. For csv_data with headers, either a string-encoded list can be used, referencing the column order of the CSV data for the associated datatable column names:

```
'[null, dt_col_nameA, null, null, dt_col_nameC, dt_col_nameB]'
```

Alternatively, a string-encoded dictionary can be used mapping CSV header names to datatable column names:


```
'{
  "hdr1": "dt_col_name1",
  "hdr2": "dt_col_name2",
  "hdr4": "dt_col_name4"
}'
```

For csv data without headers, the mapping table will contain a string-encoded list referencing the column order of the CSV data for the associated datatable column names. For example:

```
'[null, dt_col_nameA, null, null, dt_col_nameC, dt_col_nameB]'
```

Attempts are made to match the field type of the datatable. CSV data matched to **select** and **multi-select** datatables columns must contain the correct values specified for those columns. String-based date fields will be converted into epoch timestamp values based on a date format pattern (ex. '%Y-%m-%d %H:%M:%S.%f') for **datetimepicker** and **datepicker** datatable column types. See <https://strftime.org/> for the formatted values to use. Epoch date field values are also supported.

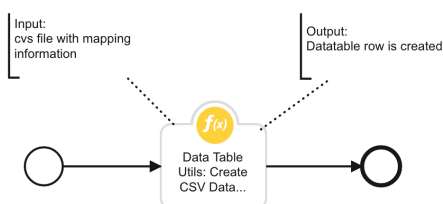
Name * Example: Data Table Utils: Create CSV Datatable

API Name *  example_create_csv_datatable

Description Take CVS data and add the results to a named datatable. Results of the function are written to an incident note.

Object Type * Attachment

Creator  Resilient Sysadmin
 Last Modified 11/05/2020 16:25
 Last Modified By  Resilient Sysadmin
 Associated Rules **Example: Create CSV Datatable**



► Inputs:

Name	Type	Required	Example	Tooltip
incident_id	number	Yes	—	—
attachment_id	number	No	—	—
dt_csv_data	text	No	CSV Data	string of cvs data consisting an optional header row followed by rows of delimiter separated data. Each delimiter separated field may contain quotes.
dt_datatable_name	text	Yes	Datatable Name	API name of datatable
dt_date_time_format	text	No	E.g. dd/mm/yyyy	If the CSV data contains date entries, provide the format for the date for conversion to a epoch timestamp.
dt_has_headers	boolean	No	—	boolean True/Yes if the csv_data contains header information to match with the column names of the datatable. If False/No, the data is added to the datatable in column order.
dt_mapping_table	text	Yes	Mapping Table	String formatted json: ""{ "column_header": "column_name", ...}"" or string formatted json list: ""[dt_colA, null, dt_colB]""
dt_start_row	number	No	—	first cvs data row to add to the datatable
dt_max_rows	number	No	—	limit the number of rows to include. No value means unlimited.

► Outputs:

```

results = {
    # TODO: Copy and paste an example of the Function Output within this
    code block.
    # To view the output of a Function, run resilient-circuits in DEBUG
    mode and invoke the Function.
    # The Function results will be printed in the logs: "resilient-
    circuits run --loglevel=DEBUG"
}

```

► Workflows

► Example Pre-Process Script:

```

# Data Table Utils: Example: CSV Table
#####
### Define Inputs ###
#####
# The ID of this incident
inputs.incident_id = incident.id
# The api name of the Data Table to update
inputs.dt_datable_name = "dt_utils_test_data_table"
# uncomment attachment_id when reading csv data from an attachmennt
##inputs.attachment_id = attachment.id

# The CSV data. Use either dt_csv_data or attachment_id
data =
u"""hdr_number,hdr_text,hdr_datetime,hdr_boolean,hdr_select,hdr_multiselect,hdr_extra
18023,"summary 中国人",6/6/20 8:12,yes,3,"a, b",中"""
data_no_headers = u"""18023,"summary 中国人",yes,6/6/20 8:12,3,"a, b",中"""
inputs.dt_csv_data = data
# A boolean to determine if CSV headers are present
inputs.dt_has_headers = True

## The mapping format should be "cvs_header":"dt_column_name"
mapping = '''{
    "hdr_number": "number",
    "hdr_text": "text",
    "hdr_boolean": "boolean",
    "hdr_datetime": "datetime",
    "hdr_select": "select",
    "hdr_multiselect": "multi_select"
}'''
# mappings of csv data without headers will be a list of data_table column
names. Use null to bypass a csv data column
mapping_no_headers =
'''["number","text","boolean","datetime","select","multi_select"]'''
inputs.dt_mapping_table = mapping
# year - %Y, month - %m, day - %d, hour - %H, minutes - %M, seconds - %S,

```

```
milliseconds - %f, timezone offset - %z'
inputs.dt_date_time_format = "%d/%m/%y %H:%M"
# optional start row csv data. The first data row = 1
#inputs.dt_start_row = 0
# optional max number of csv rows to add relative to dt_start_row
#inputs.dt_max_rows = 5
```

► Example Post-Process Script:

```
if results.success:
    note_text = u""Results from Data Table Utils: Create CSV Datatable\nData
    Source: {0}\nRows added: {1}\nRows not added: {2}"".format(
    results.content.data_source, results.content.rows_added,
    results.content.rows_with_errors )
    incident.addNote(note_text)
else:
    incident.addNote(u"Error: Failed to add rows")
```

Data Table - Example CSV Datatable

This datatable is used for testing purposes to run the example Rules and Workflows. It contains all the different datatable column types for function testing.

Example CSV Datatable

Search...

PrintExport

name	number	text	datetime	boolean	select	multi_select	
1.2.3.4	—	Done	11/05/2020 16:28:00	Unknown	4	<div>a b c d e f</div>	<div>⋮</div>
—	—	—	01/13/2020 12:30:00	Unknown	—	—	<div>⋮</div>
—	123	summary 中国人	01/02/2020 12:30:00	Yes	1	<div>a</div>	<div>⋮</div>
—	124	—	01/03/2020 12:30:00	No	2	<div>a</div>	<div>⋮</div>
—	—	ab	01/04/2020 12:30:00	Yes	3	<div>a b c</div>	<div>⋮</div>
—	126	abc	01/05/2020 12:30:00	Unknown	4	<div>a b c d</div>	<div>⋮</div>

API Name:

dt_utils_test_data_table

Columns:

Column Name	API Access Name	Type	Tooltip
name	dt_col_name	text	-
text	text	text	-
number	number	number	-

Column Name	API Access Name	Type	Tooltip
boolean	boolean	boolean	-
datetime	datetime	datetimepicker	-
select	select	select	-
multi_select	multi_select	multiselect	-

Rules

Rule Name	Object	Workflow Triggered
Get Data Table Row	artifact	example_data_table_utils_get_row
Get Data Table Rows	artifact	example_data_table_utils_get_rows
Update Data Table Row	artifact	example_data_table_utils_update_row
Update Current Row	dt_utils_test_data_table	update_row
Delete Data Table Row	artifact	example_data_table_utils_delete_row
Delete Current Row	dt_utils_test_data_table	example_data_table_utils_delete_row_from_datatable
Delete Rows by Name	dt_utils_test_data_table	example_data_table_utils_delete_rows_from_datatable
Delete Data Table Rows	artifact	

Rule Name	Object	Workflow Triggered
Example: Create CSV Datatable	attachment	example_create_csv_datatable