Datatable Utils

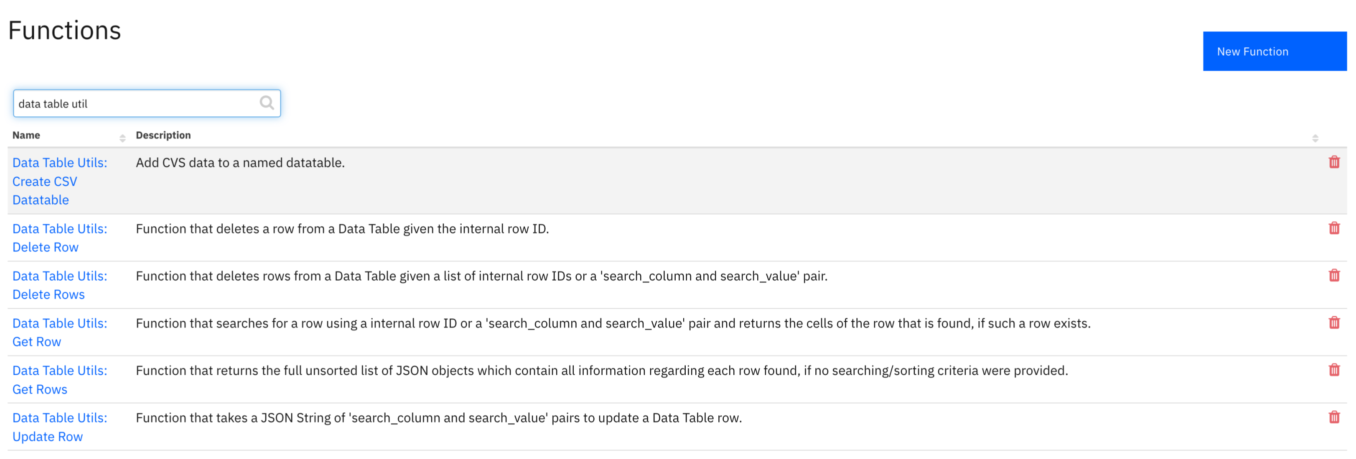
Table of Contents

Release Notes

Release	Date	Notes
v1.3.0	03/2022	Function dt_utils_get_row can now get a row from menu of a datatable row
v1.2.0	02/2021	Functions dt_utils_get_rows and dt_utils_delete_rows can now return or delete all datatable rows
v1.1.0	11/2020	Added support for App Host. New 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	2/2019	Initial Release

Overview

Functions manipulate data in a Datatable



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.

This package contains 6 functions that help you manipulate IBM SOAR Data Tables: Get Row, Get Rows, Update Row, Delete Row, Delete Rows and Convert CSV Data to a datatable.

Requirements

This app supports the IBM Security QRadar SOAR Platform and the IBM Security QRadar SOAR for IBM Cloud Pak for Security.

SOAR platform

The SOAR platform supports two app deployment mechanisms, App Host and integration server.

If deploying to a SOAR platform with an App Host, the requirements are:

- SOAR platform >= `41.0.6783`.
- The app is in a container-based format (available from the AppExchange as a `zip` file).

If deploying to a SOAR platform with an integration server, the requirements are:

- SOAR platform >= `41.0.6783`.
- The app is in the older integration format (available from the AppExchange as a `zip` file which contains a `tar.gz` file).
- Integration server is running `resilient_circuits>=33.0.0`.
- If using an API key account, make sure the account provides the following minimum permissions:

Name	Permissions
Org Data	Read
Function	Read

The following SOAR platform guides provide additional information:

- *App Host Deployment Guide*: provides installation, configuration, and troubleshooting information, including proxy server settings.
- *Integration Server Guide*: provides installation, configuration, and troubleshooting information, including proxy server settings.
- *System Administrator Guide*: provides the procedure to install, configure and deploy apps.

The above guides are available on the IBM Documentation website at ibm.biz/soar-docs. On this web page, select your SOAR platform version. On the follow-on page, you can find the *App Host Deployment Guide* or *Integration Server Guide* by expanding **Apps** in the Table of Contents pane. The System Administrator Guide is available by expanding **System Administrator**.

Cloud Pak for Security

If you are deploying to IBM Cloud Pak for Security, the requirements are:

- IBM Cloud Pak for Security >= 1.4.
- Cloud Pak is configured with an App Host.
- The app is in a container-based format (available from the AppExchange as a `zip` file).

The following Cloud Pak guides provide additional information:

- *App Host Deployment Guide*: provides installation, configuration, and troubleshooting information, including proxy server settings. From the Table of Contents, select Case Management and Orchestration & Automation > **Orchestration and Automation Apps**.
- *System Administrator Guide*: provides information to install, configure, and deploy apps. From the IBM Cloud Pak for Security IBM Documentation table of contents, select Case Management and Orchestration & Automation > **System administrator**.

These guides are available on the IBM Documentation website at ibm.biz/cp4s-docs. From this web page, select your IBM Cloud Pak for Security version. From the version-specific IBM Documentation page, select Case Management and Orchestration & Automation.

Proxy Server

The app does notsupport a proxy server.

Python Environment

Both Python 2.7 and Python 3.6 are supported. Additional package dependencies may exist for each of these packages:

- resilient-lib>=32.0.140
- resilient_circuits>=33.0.0

Installation

Install

- To install or uninstall an App or Integration on the *SOAR platform*, see the documentation at ibm.biz/soar-docs.
- To install or uninstall an App on *IBM Cloud Pak for Security*, see the documentation at ibm.biz/cp4s-docs and follow the instructions above to navigate to Orchestration and Automation.

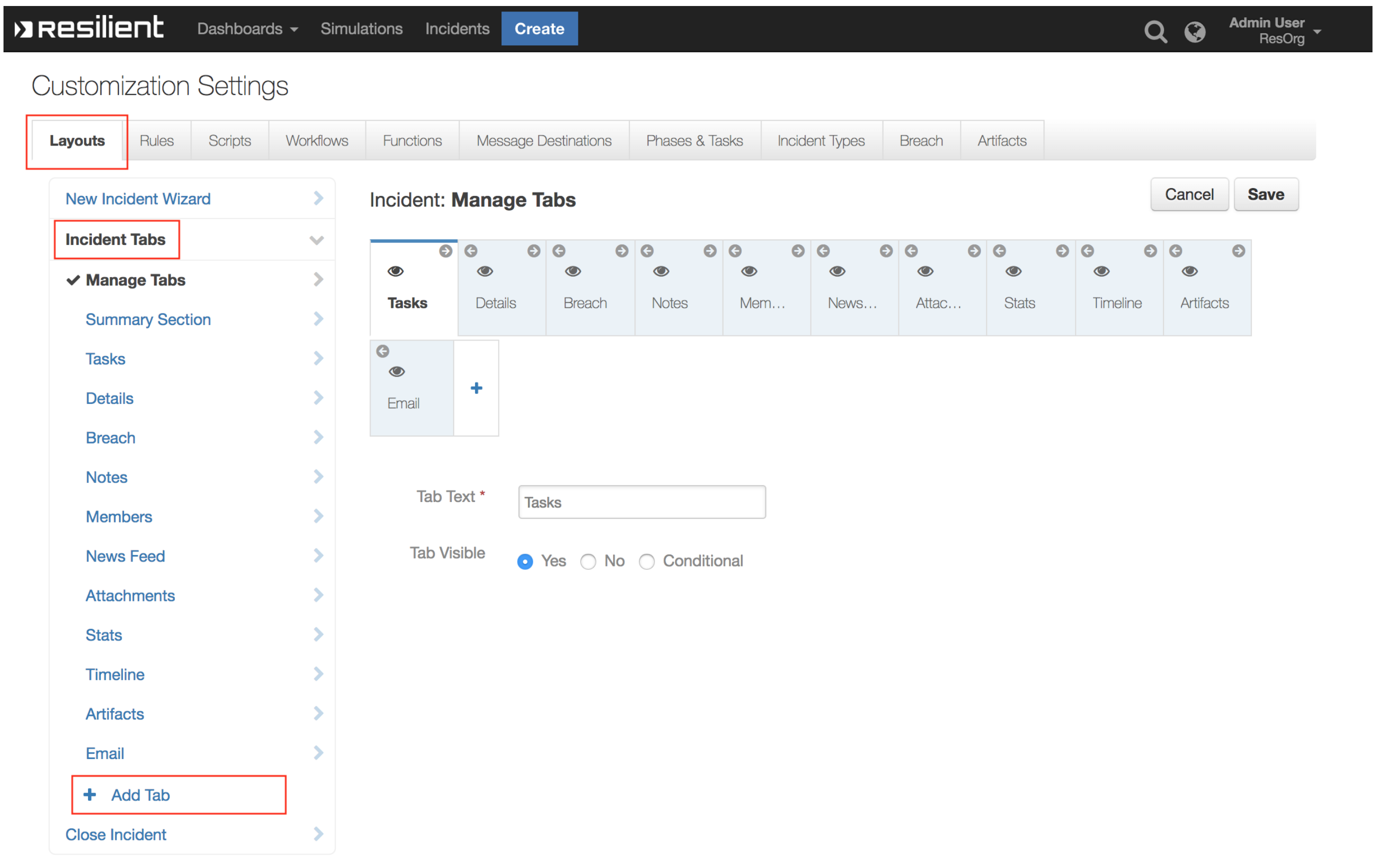
App Configuration

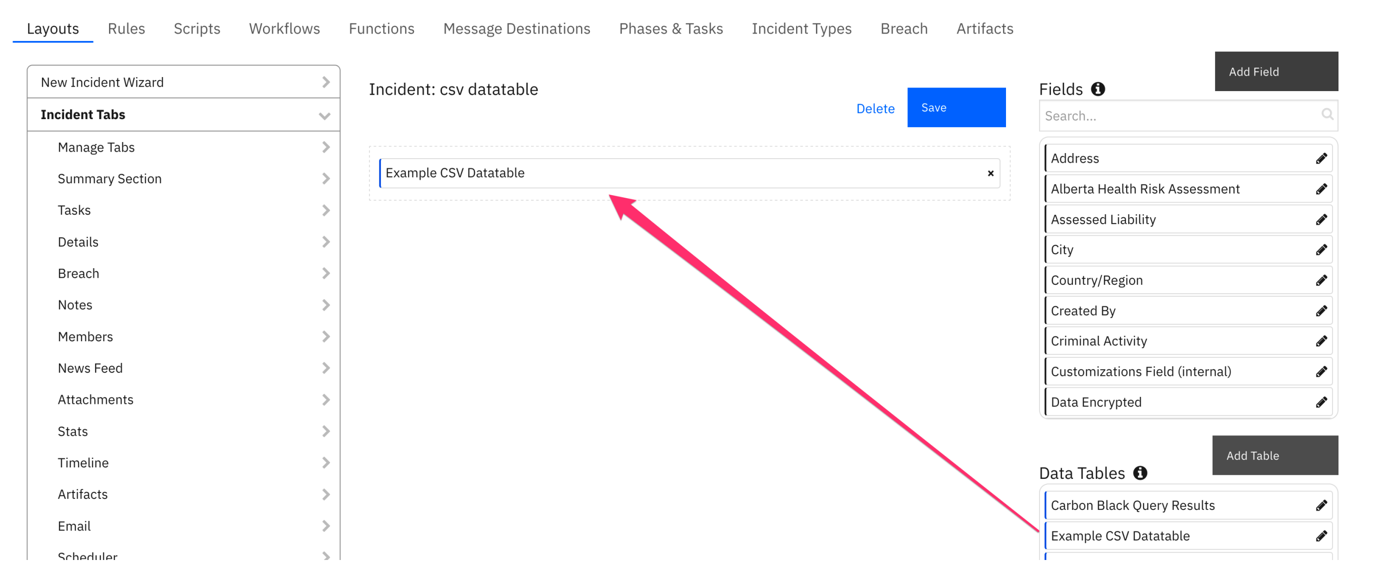
The following table provides the settings you need to configure the app. These settings are made in the app.config file. See the documentation discussed in the Requirements section for the procedure.

Config	Required	Example	Description

Setup

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





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]'
```

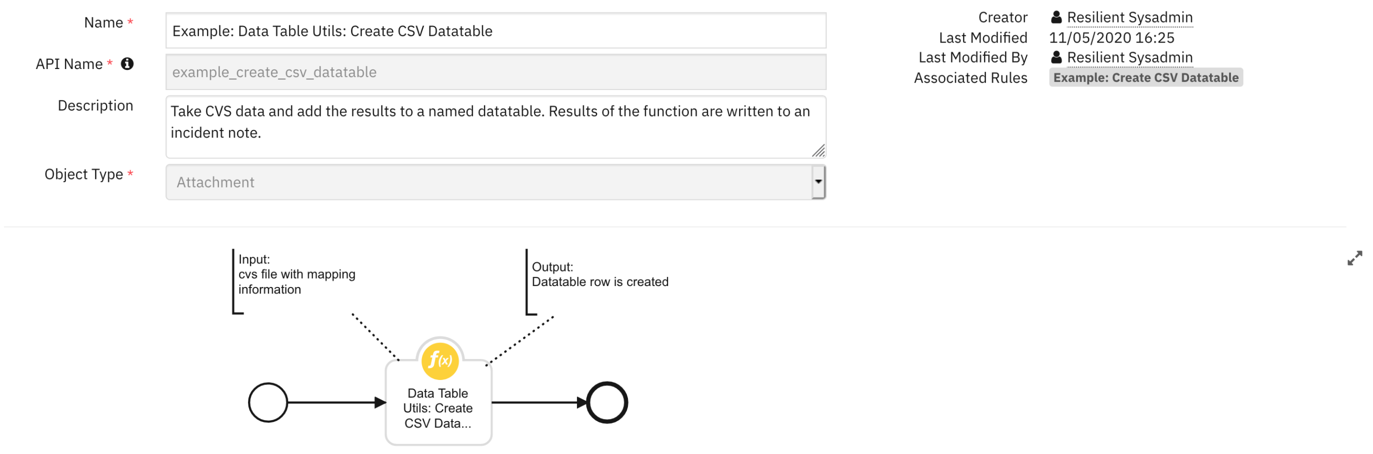
Alternatively, a string-encoded dictionary can be used mapping CSV header names to datatable column names:

```
'{
  "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

Input:
cvs file with mapping information
Output:
Datatable row is created
Data Table Utils: Create CSV Data...

▶ Inputs:

Name	Type	Required	Example	Tooltip
`attachment_id`	`number`	No	`-`	-
`dt_csv_data`	`text`	No	`CSV Data`	string of cvs data consisting an optional header row followed by rows of comma separated data. each comma separated field may contain quotes to allow for embedded commas

Name	Type	Required	Example	Tooltip
`dt_datable_name`	`text`	Yes	`Datatable Name`	string of api name of datatable
`dt_date_time_format`	`text`	No	`E.g. dd/mm/yyyy`	If you're data contains date entries, provide the format for the date
`dt_has_headers`	`boolean`	No	`-`	boolean True if the csv_data contains header information to match with the column names of the datatable. If False, the data is added to the datatable in column order.
`dt_mapping_table`	`text`	Yes	`""" {"csv_hdr1":"datatable_column_name, ...}"""`	String-encoded JSON of csv header to datatable column mappings
`dt_max_rows`	`number`	No	`-`	limit the number of rows to include
`dt_start_row`	`number`	No	`-`	Row to start adding to datatable. Use 1 if dt_has_headers = True for first data row
`incident_id`	`number`	Yes	`-`	-

▶ Outputs:

> **NOTE:** This example might be in JSON format, but `results` is a Python Dictionary on the SOAR platform.

```
results = {
  "content": {
    "data_source": "CSV data",
    "rows_added": 1,
    "rows_with_errors": 0
  },
  "inputs": {
    "dt_csv_data":
"hdr_number,hdr_text,hdr_boolean,hdr_datetime,hdr_select,hdr_multiselect,hdr_extra
\n18023,\"summary \u4e2d\u56fd\u4eba\",yes,6/6/20 8:12,3,\"a, b\",\u4e2d",
    "dt_datable_name": "dt_utils_test_data_table",
    "dt_date_time_format": "%m/%d/%y %H:%M",
    "dt_has_headers": true,
    "dt_mapping_table": "{\n  \"hdr_number\": \"number\",\n  \"hdr_text\":
\"text\",\n  \"hdr_boolean\": \"boolean\",\n  \"hdr_datetime\": \"datetime\",\n
\"hdr_select\": \"select\",\n  \"hdr_multiselect\": \"multi_select\"\n}",
    "incident_id": 2096
  },
  "metrics": {
    "execution_time_ms": 9472,
    "host": "local",
    "package": "fn-datatable-utils",
    "package_version": "1.3.0",
    "timestamp": "2022-03-30 09:35:48",
    "version": "1.0"
  },
  "raw": "{\"data_source\": \"CSV data\", \"rows_added\": 1, \"rows_with_errors\":
0}",
  "reason": null,
  "success": true,
  "version": "1.0"
}
```

▶ 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_boolean,hdr_datetime,hdr_select,hdr_multiselect,hdr_extr
a
18023,"summary 中国人",yes,6/6/20 8:12,3,"a, b",中"""
data_no_headers = u"""18023,"summary 中国人",yes,6/6/20 8:12,3,"a, b",中,x,y,z"""
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","x","y","z"]'''
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 = "%m/%d/%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:

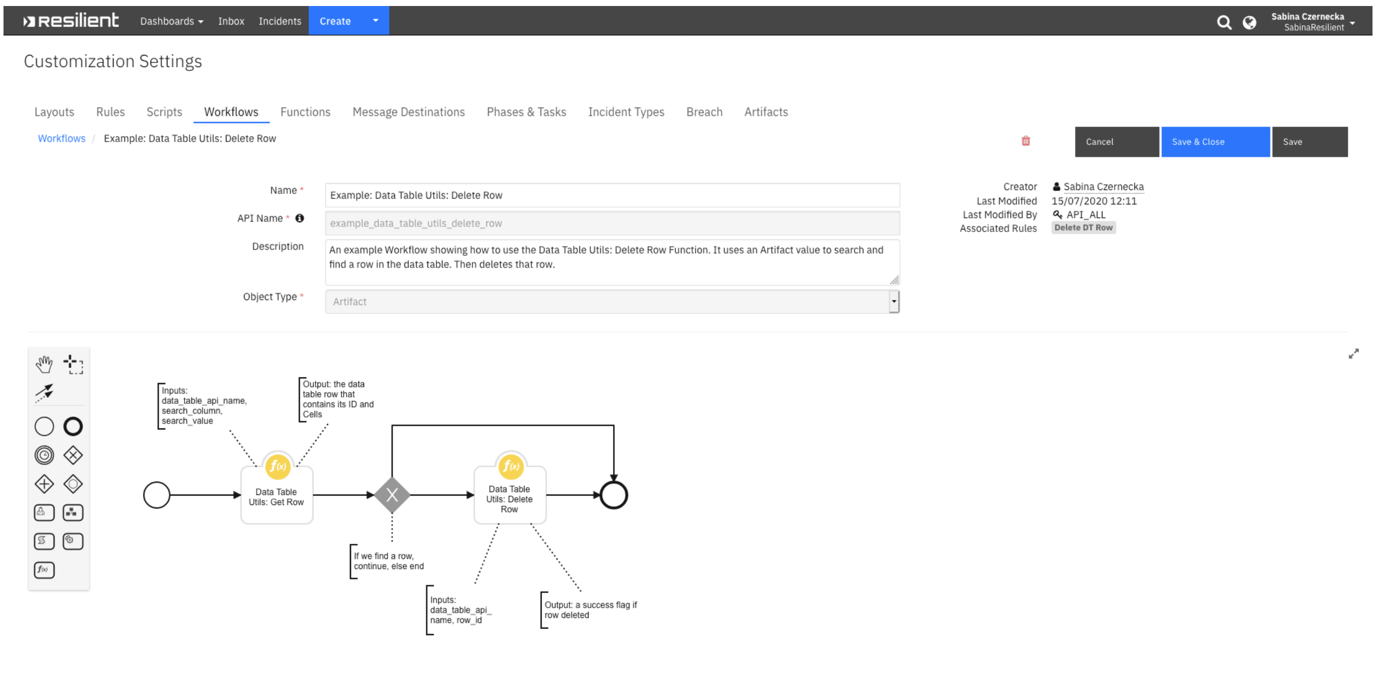
```
if results.success:
  note_text = u"""Results from Data Table Utils: Create CSV Datatable\nData Source:
{0}\nRows added: {1}\nRows not added: {2}""".format( results.content.data_source,
results.content.rows_added, results.content.rows_with_errors )
  incident.addNote(note_text)
else:
  incident.addNote(u"Error: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

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
Dashboards
Inbox
Incidents
Create
Sabina Czernecka
SabinaResilient
Layouts
Rules
Scripts
Workflows
Functions
Message Destinations
Phases & Tasks
Incident Types
Breach
Artifacts
Cancel
Save & Close
Save
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
If we find a row, continue, else end
Data Table Utils: Delete Row
Inputs: data_table_api_name, row_id
Output: a success flag if row deleted

Customization Settings

Workflows / Example: Data Table Utils: Delete Row

Name *	Example: Data Table Utils: Delete Row
API Name *	example_data_table_utils_delete_row
Description	An 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 Type *	Artifact

Creator	Sabina Czernecka
Last Modified	15/07/2020 12:11
Last Modified By	API_ALL
Associated Rules	Delete DT Row

▶ 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
`incident_id`	`number`	Yes	`-`	-

▶ Outputs:

> **NOTE:** This example might be in JSON format, but `results` is a Python Dictionary on the SOAR platform.

```
results = {
  "inputs": {
    "dt_utils_datatable_api_name": "dt_utils_test_data_table",
    "dt_utils_row_id": 1,
    "incident_id": 2096
  },
  "row": {
    "hints": [],
    "message": null,
    "success": true,
    "title": null
  },
  "success": true
}
```

▶ 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"]
```

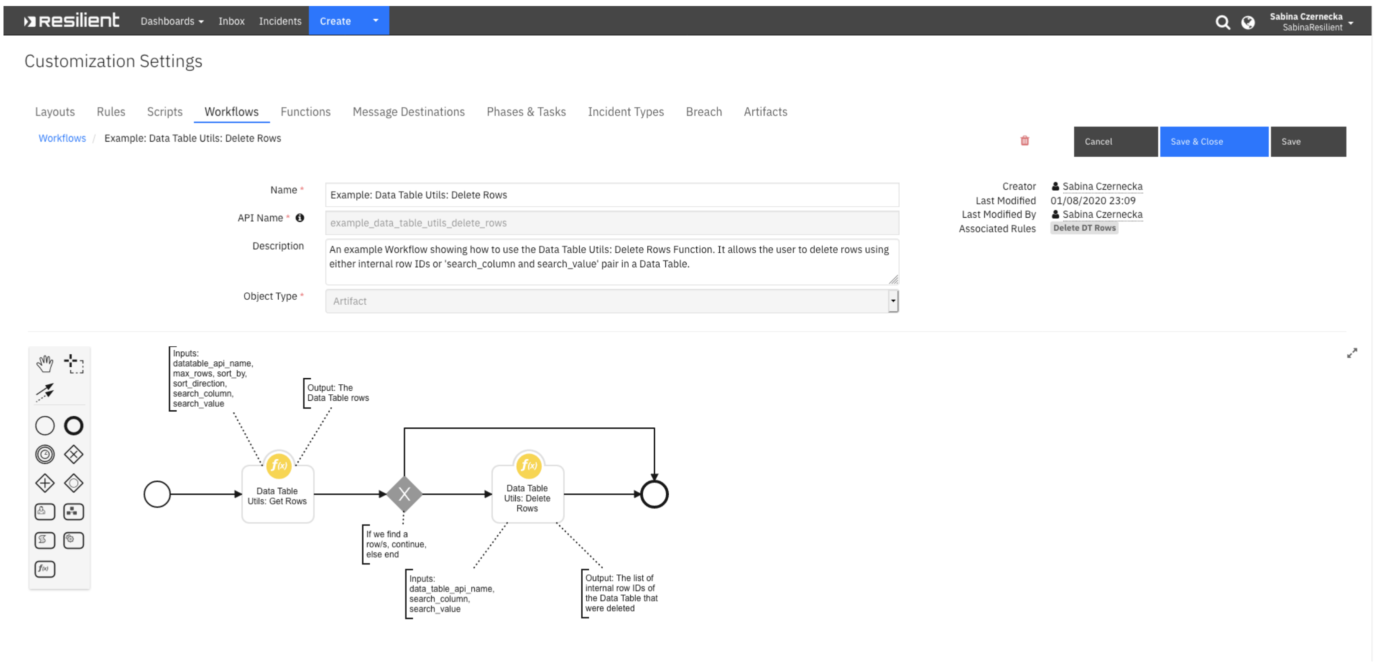
▶ Example Post-Process Script:

```
# {'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

Workflows / Example: Data Table Utils: Delete Rows

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
Data Table Utils: Get Rows
If we find a row/s, continue, else end
Data Table Utils: Delete Rows
Inputs: data_table_api_name, search_column, search_value
Output: The list of internal row IDs of the Data Table that were deleted

▶ Inputs:

Name	Type	Required	Example	Tooltip

Name	Type	Required	Example	Tooltip
`dt_utils_datatable_api_name`	`text`	Yes	`-`	The API name of the Data Table
`dt_utils_delete_all_rows`	`boolean`	No	`-`	explicitly delete all rows
`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:

> **NOTE:** This example might be in JSON format, but `results` is a Python Dictionary on the SOAR platform.

```
results = {
  "inputs": {
    "dt_utils_datatable_api_name": "dt_utils_test_data_table",
    "dt_utils_delete_all_rows": false,
    "dt_utils_rows_ids": "[5, 6]",
    "dt_utils_search_column": null,
    "dt_utils_search_value": null,
    "incident_id": 2096
  },
  "rows_ids": [
    5,
    6
  ],
  "success": true
}
```

▶ 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)
```

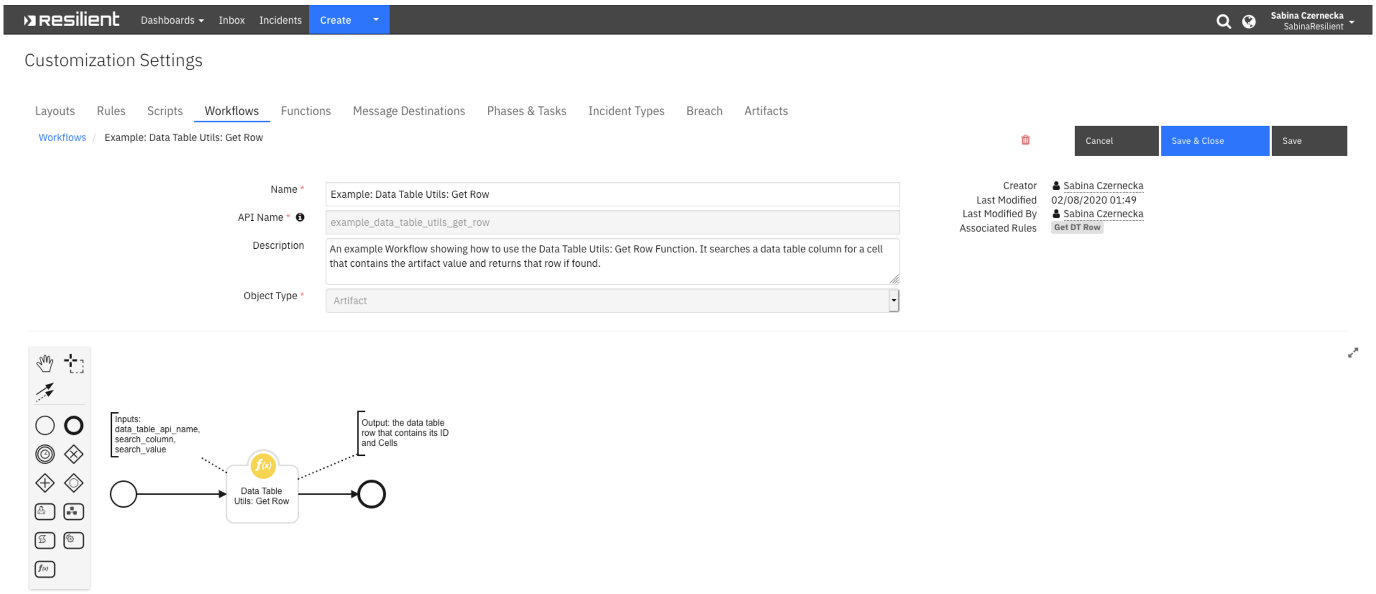
▶ Example Post-Process Script:

```
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: 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.



Customization Settings

Workflows / Example: Data Table Utils: Get Row

Name * Example: Data Table Utils: Get Row

API Name * example_data_table_utils_get_row

Description An example Workflow showing how to use the Data Table Utils: Get Row Function. It searches a data table column for a cell that contains the artifact value and returns that row if found.

Object Type * Artifact

Creator Sabina Czernecka
Last Modified 02/08/2020 01:49
Last Modified By Sabina Czernecka
Associated Rules Get DT Row

Inputs: data_table_api_name, search_column, search_value

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

▶ Inputs:

Name	Type	Required	Example	Tooltip

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:

> **NOTE:** This example might be in JSON format, but `results` is a Python Dictionary on the SOAR platform.

```
results = {
  "inputs": {
    "dt_utils_datatable_api_name": "dt_utils_test_data_table",
    "dt_utils_row_id": null,
    "dt_utils_search_column": "dt_col_name",
    "dt_utils_search_value": "something",
    "incident_id": 2096
  },
  "row": {
    "actions": [
      {
        "enabled": true,
        "id": 14,
        "name": "Delete Current Row"
      },
      {
        "enabled": true,
        "id": 17,
        "name": "Delete Rows by Name"
      },
      {
        "enabled": true,
        "id": 21,
        "name": "Update Current Row"
      }
    ],
    "cells": {
      "boolean": {
        "id": "boolean",
        "row_id": 5,
        "value": false
      },
      "datetime": {
        "id": "datetime",
        "row_id": 5,
        "value": 1647446419000
      },
      "dt_col_name": {
        "id": "dt_col_name",
        "row_id": 5,
        "value": "something"
      },
      "multi_select": {
        "id": "multi_select",
        "row_id": 5,
        "value": [
          "j",
          "g"
        ]
      },
      "number": {
        "id": "number",
        "row_id": 5,
        "value": 56
      },
      "select": {
        "id": "select",
        "row_id": 5,
        "value": "2"
      },
      "text": {
        "id": "text",
        "row_id": 5,
        "value": "fng"
      }
    },
    "id": 5,
    "inc_id": 2096,
    "inc_name": "A",
    "inc_owner": "admin@example.com",
    "table_name": "Example CSV Datatable",
    "type_id": 1000,
    "version": 1
  },
  "success": true
}
```

▶ Example Pre-Process Script:

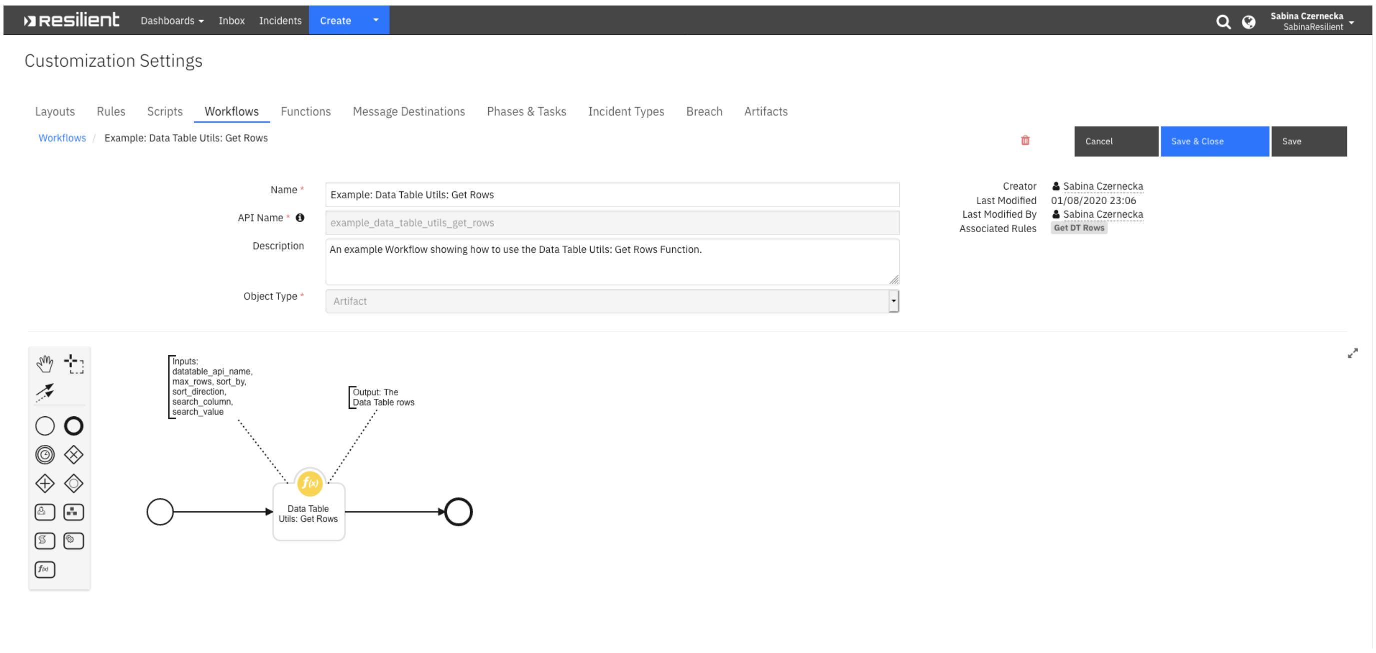
```
# Data Table Utils: Example: Get Row

#####################
### 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 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.



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
Creator	Sabina Czernecka
Last Modified	01/08/2020 23:06
Last Modified By	Sabina Czernecka
Associated Rules	Get DT Rows

▶ 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:

> **NOTE:** This example might be in JSON format, but `results` is a Python Dictionary on the SOAR platform.

```
results = {
  "inputs": {
    "dt_utils_datatable_api_name": "dt_utils_test_data_table",
    "dt_utils_max_rows": 0,
    "dt_utils_search_column": "dt_col_name",
    "dt_utils_search_value": "something",
    "dt_utils_sort_by": null,
    "dt_utils_sort_direction": "ASC",
    "incident_id": 2096
  },
  "rows": [
    {
      "actions": [
        {
          "enabled": true,
          "id": 14,
          "name": "Delete Current Row"
        },
        {
          "enabled": true,
          "id": 17,
          "name": "Delete Rows by Name"
        },
        {
          "enabled": true,
          "id": 21,
          "name": "Update Current Row"
        }
      ],
      "cells": {
        "boolean": {
          "id": "boolean",
          "row_id": 7,
          "value": false
        },
        "datetime": {
          "id": "datetime",
          "row_id": 7,
          "value": 1646110800000
        },
        "dt_col_name": {
          "id": "dt_col_name",
          "row_id": 7,
          "value": "something"
        },
        "multi_select": {
          "id": "multi_select",
          "row_id": 7,
          "value": [
            "f",
            "h"
          ]
        },
        "number": {
          "id": "number",
          "row_id": 7,
          "value": 346
        },
        "select": {
          "id": "select",
          "row_id": 7,
          "value": "4"
        },
        "text": {
          "id": "text",
          "row_id": 7,
          "value": "sfg"
        }
      },
      "id": 7,
      "inc_id": 2096,
      "inc_name": "A",
      "inc_owner": "admin@example.com",
      "table_name": "Example CSV Datatable",
      "type_id": 1000,
      "version": 1
    },
    {
      "actions": [
        {
          "enabled": true,
          "id": 14,
          "name": "Delete Current Row"
        },
        {
          "enabled": true,
          "id": 17,
          "name": "Delete Rows by Name"
        },
        {
          "enabled": true,
          "id": 21,
          "name": "Update Current Row"
        }
      ],
      "cells": {
        "boolean": {
          "id": "boolean",
          "row_id": 8
        },
        "datetime": {
          "id": "datetime",
          "row_id": 8,
          "value": 1646357232000
        },
        "dt_col_name": {
          "id": "dt_col_name",
          "row_id": 8,
          "value": "something"
        },
        "multi_select": {
          "id": "multi_select",
          "row_id": 8,
          "value": [
            "d"
          ]
        },
        "number": {
          "id": "number",
          "row_id": 8,
          "value": 89
        },
        "select": {
          "id": "select",
          "row_id": 8,
          "value": "5"
        },
        "text": {
          "id": "text",
          "row_id": 8,
          "value": "fghsn"
        }
      },
      "id": 8,
      "inc_id": 2096,
      "inc_name": "A",
      "inc_owner": "admin@example.com",
      "table_name": "Example CSV Datatable",
      "type_id": 1000,
      "version": 1
    }
  ],
  "success": true
}
```

▶ 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:

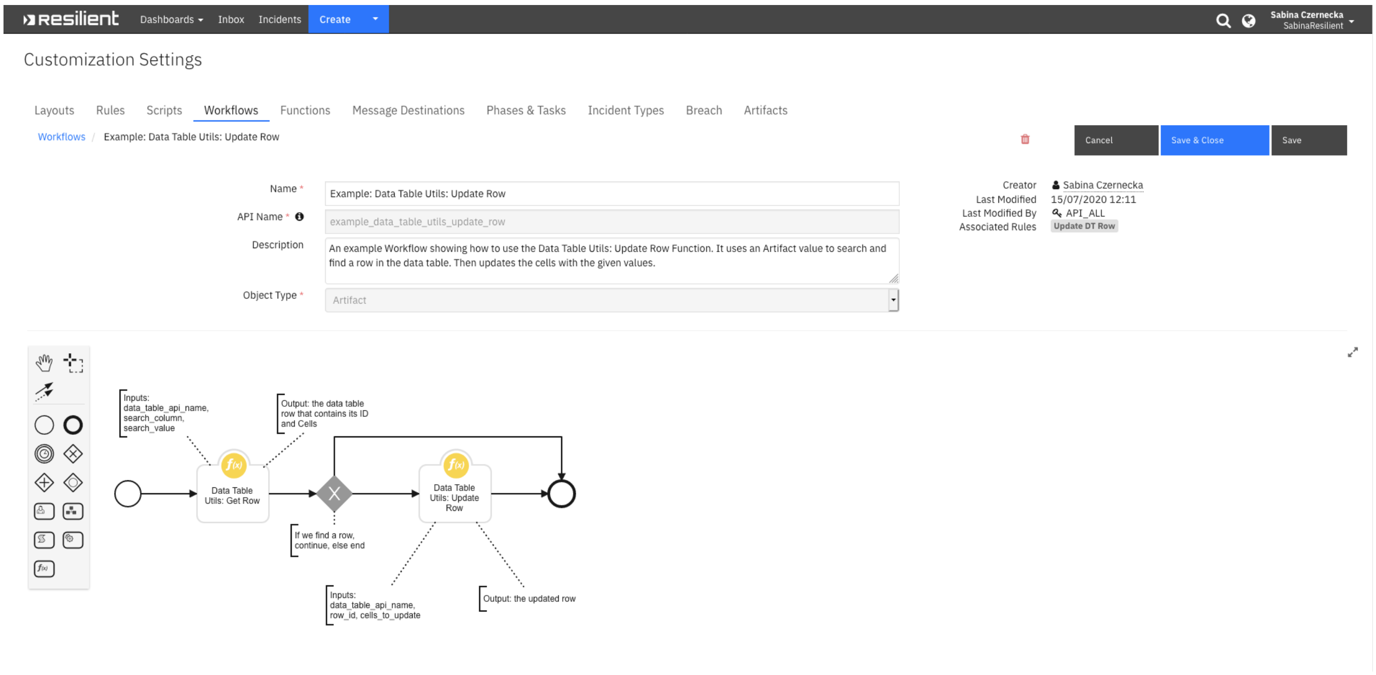
```
if not results.success:
  incident.addNote(helper.createRichText("<b>Result from Example: Data Table Utils:
Delete Rows</b><br>No rows found."))
```

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
Dashboards
Inbox
Incidents
Create
Sabina Czernecka
SabinaResilient
Customization Settings
Layouts
Rules
Scripts
Workflows
Functions
Message Destinations
Phases & Tasks
Incident Types
Breach
Artifacts
Workflows / Example: Data Table Utils: Update 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
Data Table Utils: Get Row
If we find a row, continue, else end
Data Table Utils: Update Row
Inputs: data_table_api_name, row_id, cells_to_update
Output: the updated row

Name * Example: Data Table Utils: Update Row

API Name * example_data_table_utils_update_row

Description An 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 Type * Artifact

Creator Sabina Czernecka
Last Modified 15/07/2020 12:11
Last Modified By API_ALL
Associated Rules Update DT 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

Name	Type	Required	Example	Tooltip
`dt_utils_row_id`	`number`	No	`-`	The internal ID of the row to be retrieved
`incident_id`	`number`	Yes	`-`	-

▶ Outputs:

> **NOTE:** This example might be in JSON format, but `results` is a Python Dictionary on the SOAR platform.

```
results = {
  "inputs": {
    "dt_utils_cells_to_update": {
      "datetime": 1648646740137,
      "text": "Done"
    },
    "dt_utils_datatable_api_name": "dt_utils_test_data_table",
    "dt_utils_row_id": 5,
    "incident_id": 2096
  },
  "row": {
    "actions": [
      {
        "enabled": true,
        "id": 14,
        "name": "Delete Current Row"
      },
      {
        "enabled": true,
        "id": 17,
        "name": "Delete Rows by Name"
      },
      {
        "enabled": true,
        "id": 21,
        "name": "Update Current Row"
      }
    ],
    "cells": {
      "boolean": {
        "id": "boolean",
        "row_id": 5,
        "value": false
      },
      "datetime": {
        "id": "datetime",
        "row_id": 5,
        "value": 1648646740137
      },
      "dt_col_name": {
        "id": "dt_col_name",
        "row_id": 5,
        "value": "something"
      },
      "multi_select": {
        "id": "multi_select",
        "row_id": 5,
        "value": [
          "j",
          "g"
        ]
      },
      "number": {
        "id": "number",
        "row_id": 5,
        "value": 56
      },
      "select": {
        "id": "select",
        "row_id": 5,
        "value": "2"
      },
      "text": {
        "id": "text",
        "row_id": 5,
        "value": "Done"
      }
    },
    "id": 5,
    "inc_id": 2096,
    "inc_name": "A",
    "inc_owner": "admin@example.com",
    "table_name": "Example CSV Datatable",
    "type_id": 1000,
    "version": 2
  },
  "success": true
}
```

▶ Example Pre-Process Script:

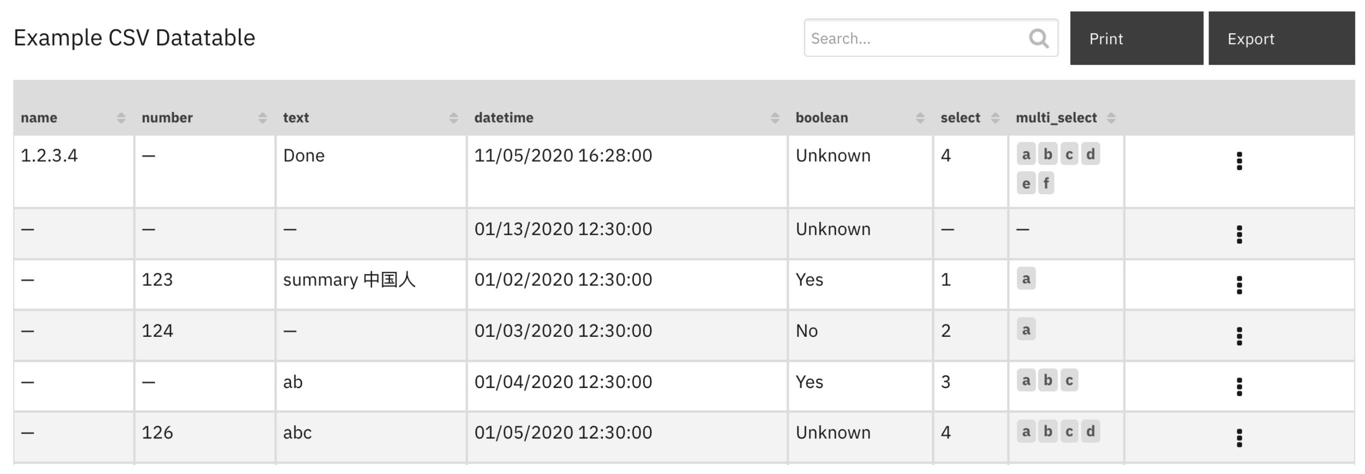
```
# Data Table Utils: Example: Update Row
import java.util.Date as Date

#######################################
### 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
```

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... Print Export

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

API Name:

dt_utils_test_data_table

Columns:

Column Name	API Access Name	Type	Tooltip
boolean	`boolean`	`boolean`	-
datetime	`datetime`	`datetimepicker`	-
multi_select	`multi_select`	`multiselect`	-
name	`dt_col_name`	`text`	-
number	`number`	`number`	-
select	`select`	`select`	-
text	`text`	`text`	-

Rules

Rule Name	Object	Workflow Triggered
Delete Current Row	dt_utils_test_data_table	`example_data_table_utils_delete_row_from_datatable`
Delete Data Table Row	artifact	`example_data_table_utils_delete_row`
Delete Data Table Rows	artifact	`example_data_table_utils_delete_rows`
Delete Rows by Name	dt_utils_test_data_table	`example_data_table_utils_delete_rows_from_datatable`
Example: Create CSV Datatable	attachment	`example_create_csv_datatable`
Get Data Table Row	artifact	`example_data_table_utils_get_row`

Rule Name	Object	Workflow Triggered
Get Data Table Rows	artifact	`example_data_table_utils_get_rows`
Update Current Row	dt_utils_test_data_table	`update_row`
Update Data Table Row	artifact	`example_data_table_utils_update_row`

Troubleshooting & Support

Refer to the documentation listed in the Requirements section for troubleshooting information.

For Support

This is a IBM Community provided App. Please search the Community ibm.biz/soarcommunity for assistance.