IBM Resilient - Data Table Helper Functions

Table of Contents

This package contains 5 functions that help you manipulate IBM Resilient Data Tables
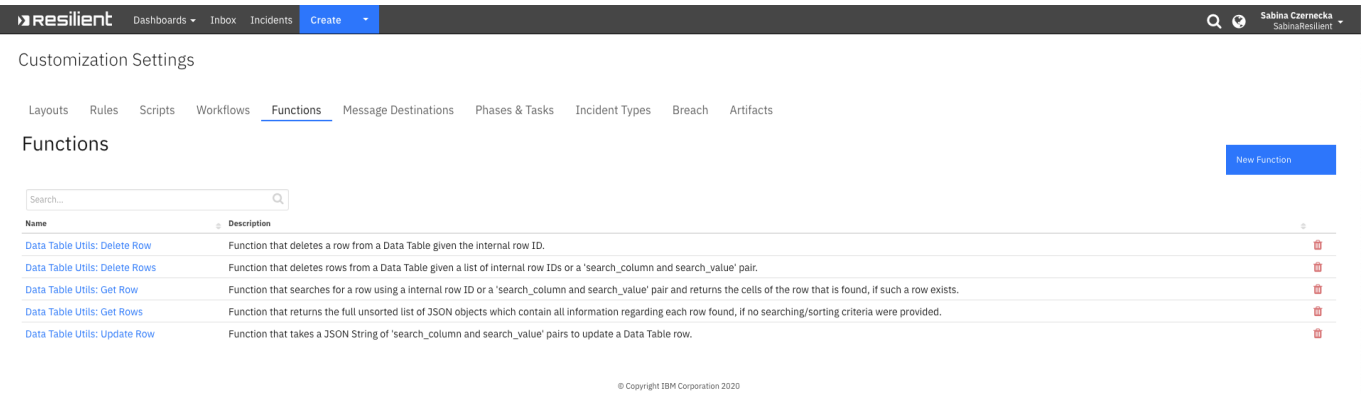
The 4 functions allow you to GET, UPDATE, DELETE a row and GET and DELETE rows in a Data Table

Function - 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.

Inputs:

Name	Type	Required	Example	Info
incident_id	Number	Yes	2095	ID of the current Incident
dt_utils_datatable_api_name	String	Yes	"dt_utils_test_data_table"	API name of the Data Table to search
dt_utils_row_id	Number	No	5	ID of the Row to get
dt_utils_search_column	String	No	"dt_col_name"	API name of the column to search
dt_utils_search_value	String	No	"Mary Murphy"	Cell value to search for

Output:

```
results = {
    'inputs': {
        'incident_id': 2095,
        'dt_utils_row_id': None,
        'dt_utils_search_column': 'dt_col_name',
        'dt_utils_search_value': 'Mary Murphy',
        'dt_utils_datatable_api_name': 'dt_utils_test_data_table'
    },
    'success': True,
    'row': {
        'type_id': 1000,
        'cells': {
            'dt_col_name': {
                'row_id': 1,
                'id': 'dt_col_name',
                'value': 'Mary Murphy'
            },
            'dt_col_status': {
                'row_id': 1,
                'id': 'dt_col_status',
                'value': 'Incomplete'
            },
            'dt_col_time': {
                'row_id': 1,
                'id': 'dt_col_time',
                'value': '10:45'
            },
            'dt_col_id': {
                'row_id': 1,
                'id': 'dt_col_id',
                'value': '635'
            }
        },
        'actions': [],
        'inc_owner': 'admin@res.com',
        'version': 2,
        'table_name': 'Test Data Table',
        'inc_name': 'Test DT Utils',
        'inc_id': 2095,
        'id': 1
    }
}
```

Pre-Process Script:

This example uses the Artifact Value as the value to search the data table for

```
# 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
```

Function - 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.

Inputs:

Name	Type	Required	Example	Tooltip
incident_id	Number	Yes	2095	ID of the current Incident
dt_utils_datatable_api_name	String	Yes	"dt_utils_test_data_table"	API name of the Data Table to search

Name	Type	Required	Example	Tooltip
dt_utils_sort_by	String	No	"dt_col_status"	The name of the API column
dt_utils_sort_direction	Select	No	{'name': 'ASC'}	-
dt_utils_max_rows	Number	No	1	A number of max rows to return
dt_utils_search_column	String	No	"dt_col_name"	API name of the column to search
dt_utils_search_value	String	No	"Mary Murphy"	Cell value to search for

Output:

```
results={
  'inputs': {
    'incident_id': 2095,
    'dt_utils_datatable_api_name': 'dt_utils_test_data_table',
    'dt_utils_sort_by': 'dt_col_status',
    'dt_utils_sort_direction': 'ASC',
    'dt_utils_max_rows': 1,
    'dt_utils_search_column': 'dt_col_name',
    'dt_utils_search_value': 'Mary Murphy'
  },
  'success': True,
  'rows': [
    {
      'type_id': 1000,
      'cells': {
        'dt_col_name': {
          'row_id': 1,
          'id': 'dt_col_name',
          'value': 'Mary Murphy'
        },
        'dt_col_status': {
```

```
            'row_id': 1,
            'id': 'dt_col_status',
            'value': 'Incomplete'
          },
          'dt_col_time': {
            'row_id': 1,
            'id': 'dt_col_time',
            'value': '10:45'
          },
          'dt_col_id': {
            'row_id': 1,
            'id': 'dt_col_id',
            'value': '635'
          }
        },
        'actions': [

        ],
        'inc_owner': 'admin@res.com',
        'version': 2,
        'table_name': 'Test Data Table',
        'inc_name': 'Test DT Utils',
        'inc_id': 2095,
        'id': 1
      }
    ]
}
```

Pre-Process Script:

This example uses the Artifact Value as the value to search the data table for

```
# 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
inputs.dt_utils_sort_direction = rule.properties.dt_utils_sort_direction

# 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
```

Post-Processing Script

```
noteText = str(results["rows"])
incident.addNote(noteText)
```

Function - Update Row:

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

Inputs:

Name	Type	Required	Example	Info
incident_id	Number	Yes	2095	ID of the current Incident

Name	Type	Required	Example	Info
dt_utils_datatable_api_name	String	Yes	"dt_utils_test_data_table"	API name of the Data Table to search
dt_utils_row_id	Number	Yes	5	ID of the Row to update
dt_utils_cells_to_update	JSON String	Yes	'{"dt_col_status":"Done", "dt_col_time":"Just Now"}'	A JSON String of column name and cell value pairs to update

Output:

```
results = {
    'inputs': {
        'incident_id': 2095,
        'dt_utils_row_id': 1,
        'dt_utils_cells_to_update': {
            'dt_col_status': 'Done',
            'dt_col_time': 'Just Now'
        },
        'dt_utils_datatable_api_name': 'dt_utils_test_data_table'
    },
    'success': True,
    'row': {
        'type_id': 1000,
        'cells': {
            'dt_col_name': {
                'row_id': 1,
                'id': 'dt_col_name',
                'value': 'Mary Murphy'
            },
            'dt_col_status': {
                'row_id': 1,
                'id': 'dt_col_status',
                'value': 'Done'
            },
            'dt_col_time': {
```

```
                      'row_id': 1,
                      'id': 'dt_col_time',
                      'value': 'Just Now'
                },
                'dt_col_id': {
                      'row_id': 1,
                      'id': 'dt_col_id',
                      'value': '635'
                }
          },
          'actions': [],
          'inc_owner': 'admin@res.com',
          'version': 3,
          'table_name': 'Test Data Table',
          'inc_name': 'Test DT Utils',
          'inc_id': 2095,
          'id': 1
    }
}
```

Pre-Process Script:

In this example we make use of the **dict_to_json_str** function to allow us to easily form a JSON String to pass column name and cell value pairs to the function in order to update the Data Table

```python
######################################
### 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))
```

```python
        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), '}')

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

# 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 =
workflow.properties.row_to_update.inputs.dt_utils_datatable_api_name

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

# The column api names and the value to update the cell to
inputs.dt_utils_cells_to_update = dict_to_json_str({
    "dt_col_time": "Just Now",
    "dt_col_status": "Done"
})
```

Function - Delete Row:

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

Inputs:

Name	Type	Required	Example	Info
incident_id	Number	Yes	2095	ID of the current Incident
dt_utils_datatable_api_name	String	Yes	"dt_utils_test_data_table"	API name of the Data Table to search
dt_utils_row_id	Number	Yes	5	ID of the Row to delete

Output:

```
results = {
    'inputs': {
        'incident_id': 2095,
        'dt_utils_row_id': 2,
        'dt_utils_datatable_api_name': 'dt_utils_test_data_table'
    },
    'success': True,
    'row': {
        'message': None,
```

```
        'hints': [],
        'success': True,
        'title': None
    }
}
```

Pre-Process Script:

```python
# 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"]
```

Function - 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.

Note: This function works with no issue when run on an Artifact workflow. There is an issue when the function is run on a custom workflow on a Data Table. If the delete rows action is performed of the row that meets the condition specified in the pre-process script, this row will not be 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	–	-

Output:

```
results = [30,31]
```

Pre-Process Script:

```python
# 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)

## Alternatively you can delete row/rows by defining the list of internal
row IDs or by defining the 'search_column and search_value' pair:
# inputs.dt_utils_rows_ids = "[2,3]"  # or
# inputs.dt_utils_search_column = "dt_col_name"
# inputs.dt_utils_search_value = "Mary Murphy"
```

Post-Process Script:

```
incident.addNote("Rows deleted: {0}".format(str(results["rows_ids"])))
```

Rules

Rule Name	Object Type	Workflow Triggered
Get DT Row	Artifact	Example: Data Table Utils: Get Row
Get DT Rows	Artifact	Example: Data Table Utils: Get Rows
Update DT Row	Artifact	Example: Data Table Utils: Update Row
Delete DT Row	Artifact	Example: Data Table Utils: Delete Row
Delete DT Rows	Artifact	Example: Data Table Utils: Delete Rows

Data table

Data Table Utils: Test Data Table

API Name:

dt_utils_test_data_table

Columns:

Column Name	API Access Name	Type
ID	dt_col_id	Text
Time	dt_col_time	Text
Name	dt_col_name	Text

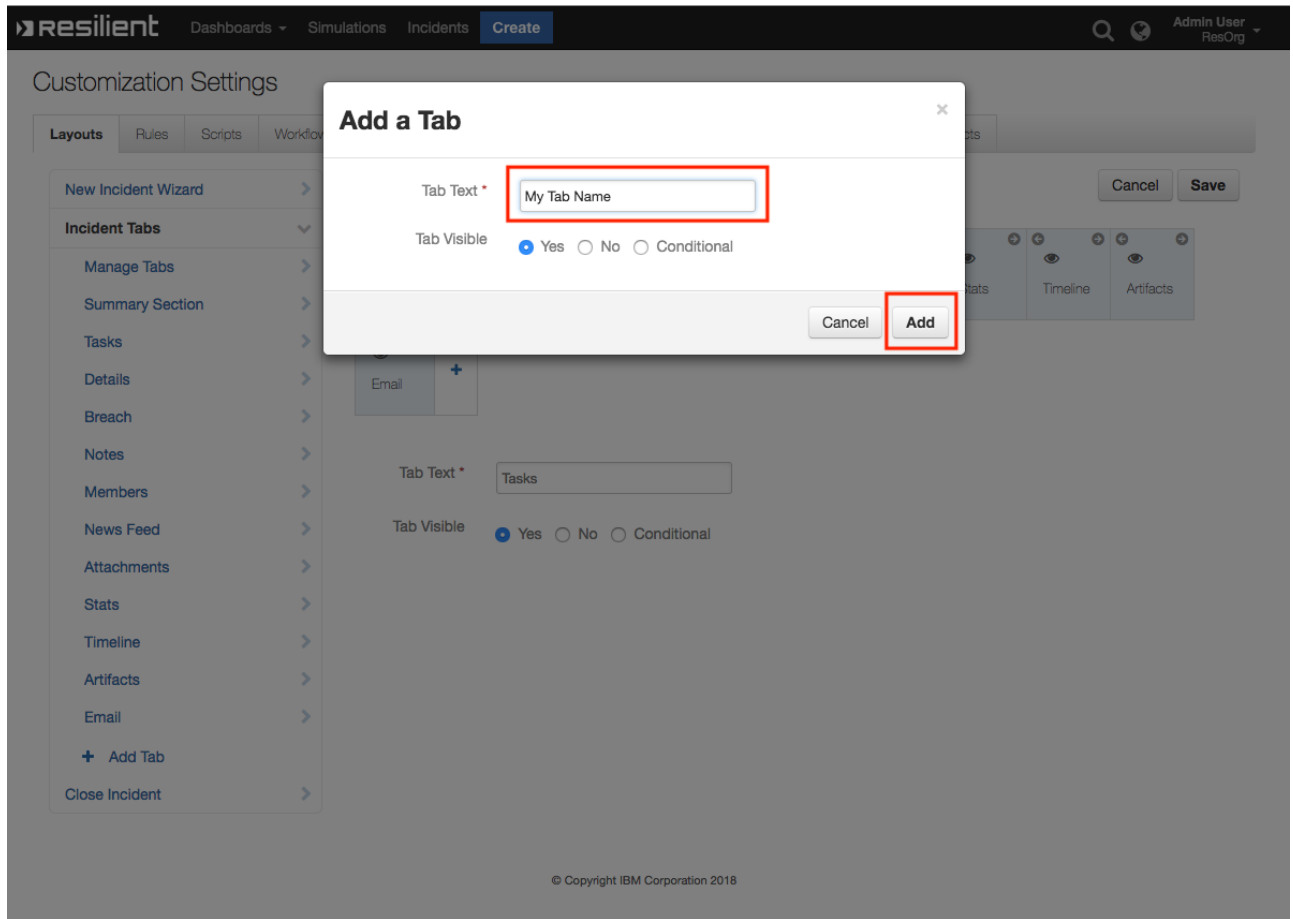| Column Name	API Access Name	Type
Status	dt_col_status	Text

Display the Datatable in an Incident

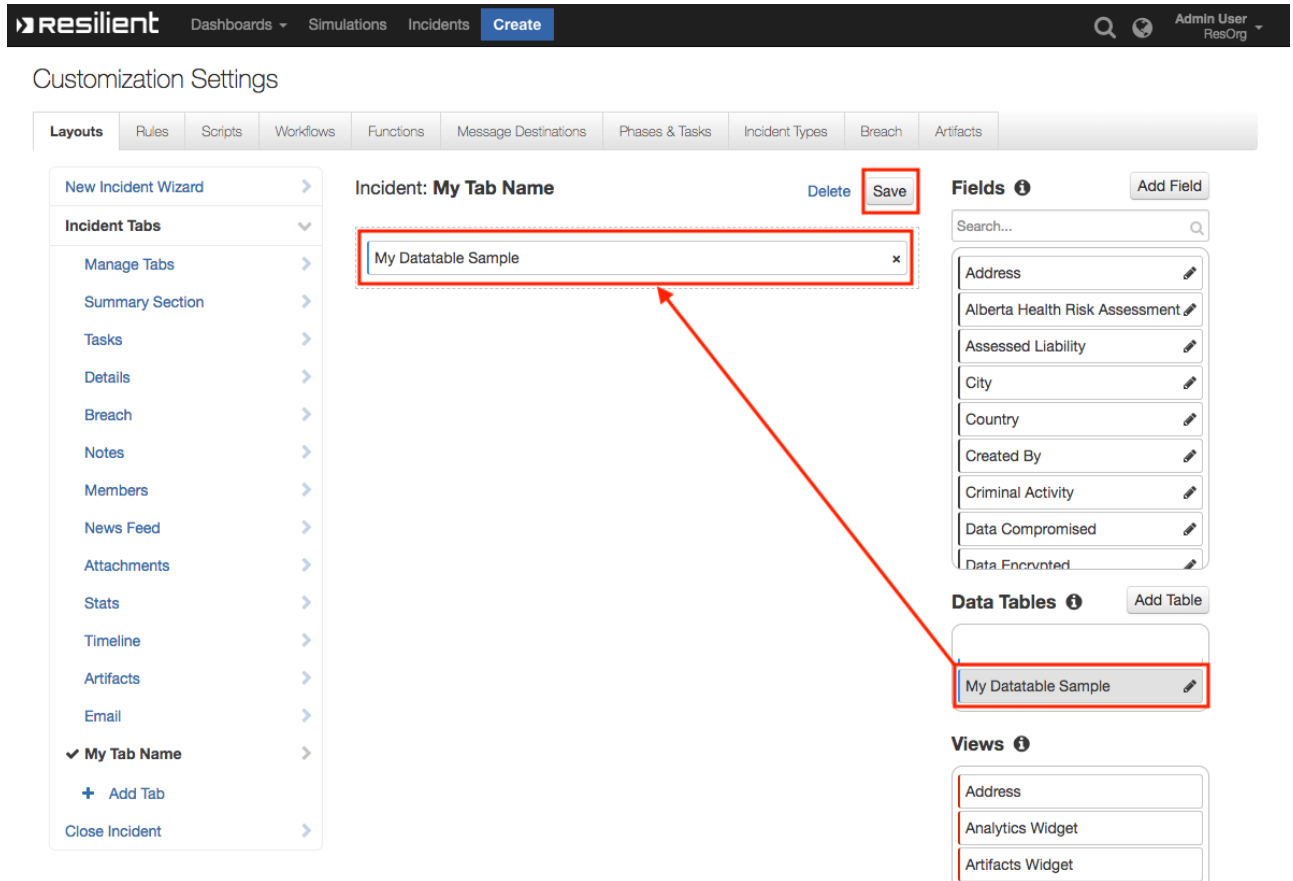- In order to **display** the Test Data Table in your Incident, you must **modify your Layout Settings**

1. Go to **Customization Settings** > **Layouts** > **Incident Tabs** > **+ Add Tab**
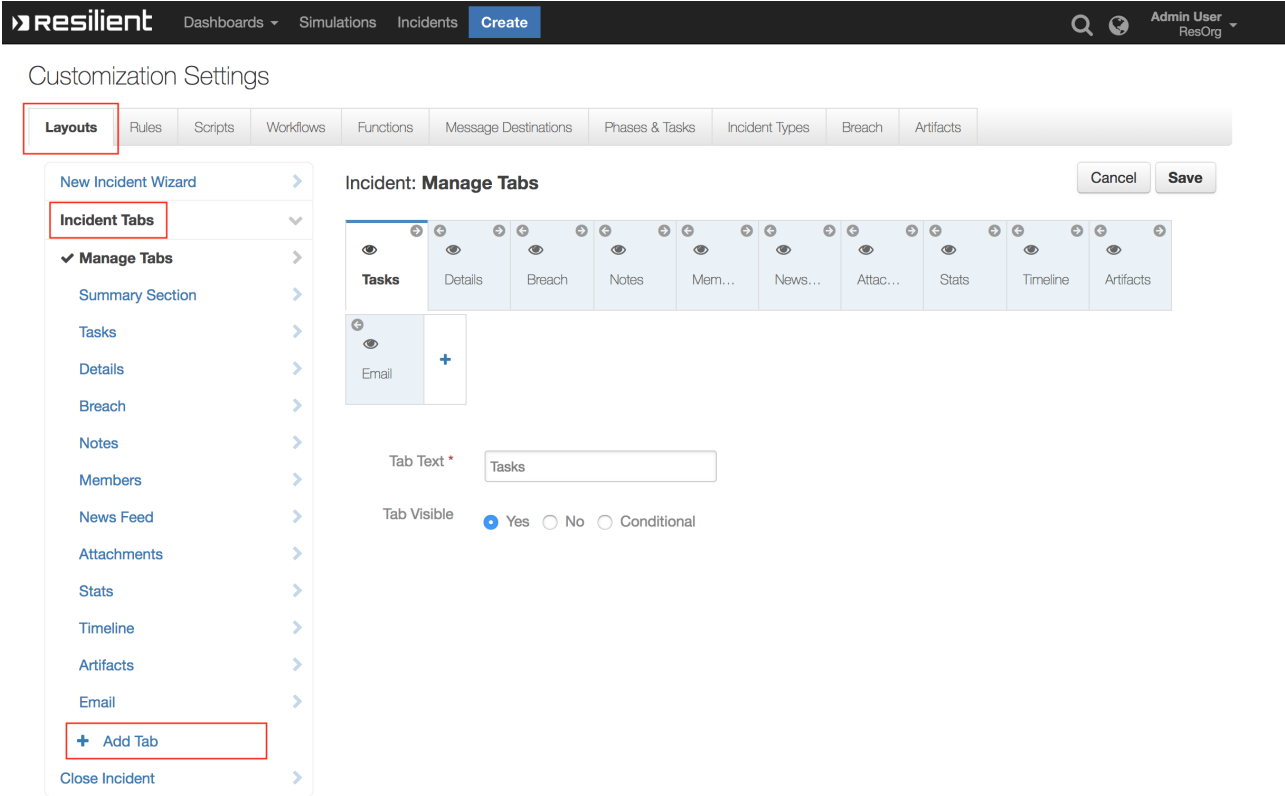
2. Enter **Tab Text**: `My Test Tab` and click **Add**

3. **Drag** the Data table into the middle and click **Save**

4. Create a new Incident and you will now see the **My Test Tab** with the **Test Data Table**