User Guide: fn_scheduler

Table of Contents

History

Version	Date	Notes
1.0.2	Sept. 2020	Added ability to use PostgreSQL DB
1.0.1	May 2020	App Host support added
1.0.0	Nov. 2019	Initial Publication

Migrating to v1.0.2

When migrating to v1.0.2 from a previous release, add the following setting to your `[fn_scheduler]` app.config section:

```
# db url if using a postgreSQL DB. Use this with AppHost
#db_url=postgresql+psycopg2://username:password@host:port/database
```

Use this setting rather than the SQLite `datastore_dir` setting to persist the scheduler DB in PostgreSQL. This is necessary in an App Host environment to retain your schedules outside the app container.

Key Features

This package of functions allows an enterprise to schedule a rule to run in the future associated with a incident, task, artifact, and datatable. Schedule times to run can be specified in the following ways:

1. cron (ex. * 0 * * * for every night at midnight)
2. interval (ex. 5h for every 5 hours, 2d for every 2 days. Valid values are s(econd), m(inute), h(our), d(ay), w(eek), M(onth))
3. date (ex. 2019/10/23 12:00:00 or 2019-10-23 12:00:00)
4. delta (ex. 1h for one hour in the future, the same values as interval are supported)

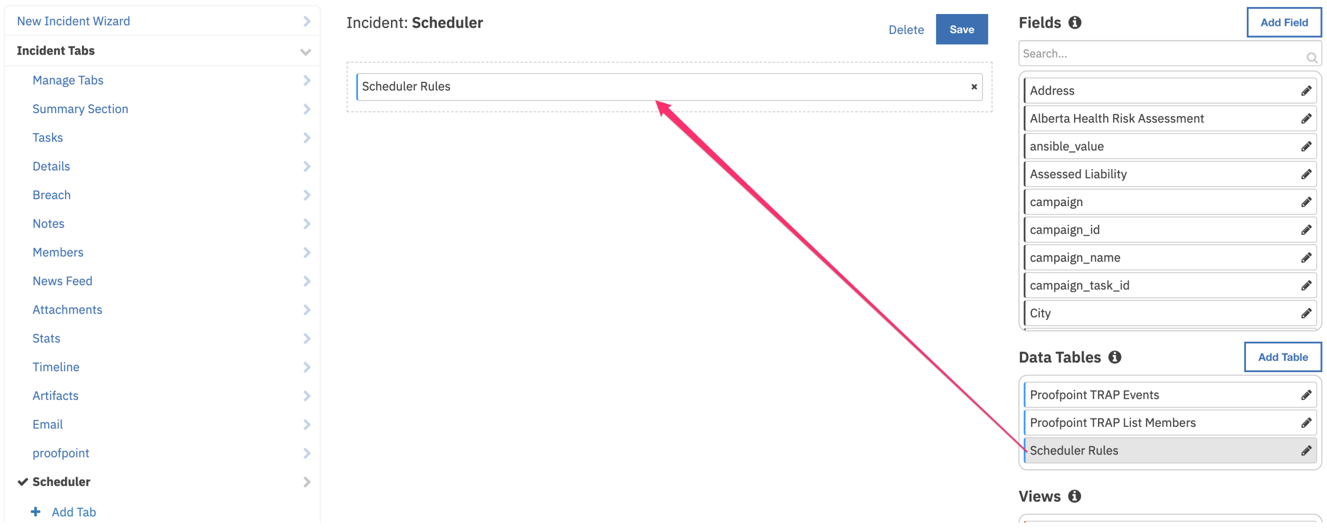
Scheduled rules using `cron` and `interval` are reoccurring whereas `date` and `delta` are single event schedules. Scheduled rules are persisted so that restarts of resilient-circuits will resume already scheduled rules.

Functions available include:

- Create a scheduled rule
- List scheduled rules
- Pause and resume scheduled rules
- Remove a scheduled rule

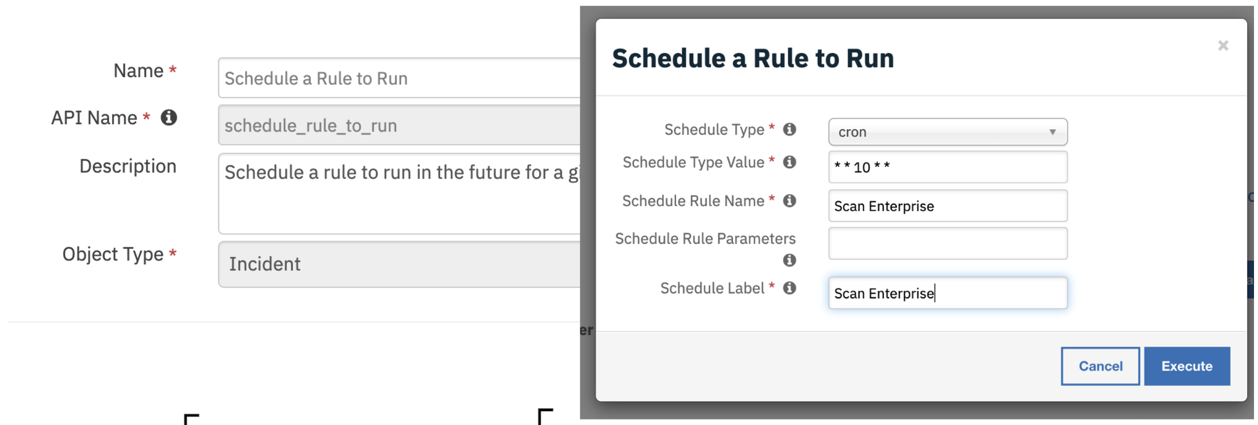
Layout

A datatable is used to display scheduled rules and to take actions such as pause, resume and remove a rule. This datatable can be added to your incident layout by adding a new tab and by dragging the `Scheduler Rules` datatable to the new tab. Remember to save the layout change.



Function - Create a Scheduled Rule

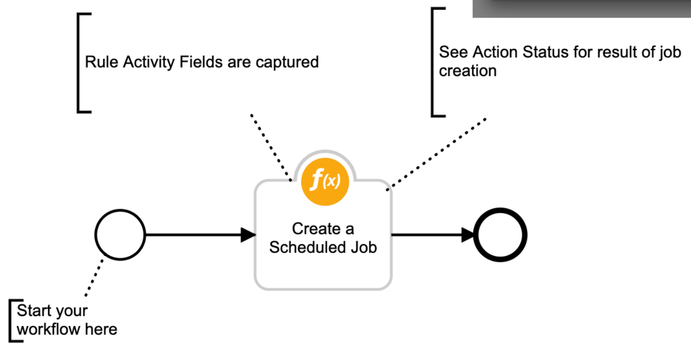
Schedule a rule to run on a schedule. This rule will be executed for a given incident, artifact, task, etc.



Name *	Schedule a Rule to Run
API Name *	schedule_rule_to_run
Description	Schedule a rule to run in the future for a g
Object Type *	Incident

Schedule a Rule to Run

Schedule Type *	cron
Schedule Type Value *	* * 10 * *
Schedule Rule Name *	Scan Enterprise
Schedule Rule Parameters	
Schedule Label *	Scan Enterprise


Rule Activity Fields are captured
See Action Status for result of job creation
Create a Scheduled Job
Start your workflow here

▶ Inputs:

Name	Type	Required	Example	Tooltip
`incident_id`	`number`	Yes	`-`	Incident Id where the rule will be executed
`object_id`	`number`	No	`-`	Id for task, artifact, attachment, datatable, etc.
`row_id`	`number`	No	`-`	row information for datatable rules
`scheduler_label_prefix`	`text`	Yes	`-`	Label to recall the created schedule. The incident id is appended to the name for uniqueness
`scheduler_rule_name`	`text`	Yes	`-`	Name of rule to schedule
`scheduler_rule_parameters`	`text`	No	`-`	Optional parameters for the rule in field=value format separated by semicolons. These fields should match the api name for the rule's activity fields
`scheduler_type`	`select`	No	`-`	type of schedule to create. cron, interval, date or delta
`scheduler_type_value`	`text`	Yes	`-`	interval, date (yyyy/mm/dd hh:mm:ss), cron or delta value

▶ Outputs:

```
results = {
  'success': True,
  'content': {
    'args': (2219, # incident_id
    None, # object_id
    None, # row_id
    u'rule3', # Rule to execute
    u'Delete rule3', # Scheduled rule Label
    49, # rule_id
    0, # object_type_id
    None,
    None),
    'executor': 'default',
    'max_instances': 1,
    'func':
'fn_scheduler.components.create_a_scheduled_rule:triggered_job',
    'id': u'rule3',
    'next_run_time': 'Oct 03 2019 12:35PM',
    'name': 'triggered_job',
    'misfire_grace_time': 1,
    'trigger': None,
    'coalesce': False,
    'version': 1,
    'kwargs': {

    }
  },
```

▶ Example Pre-Process Script:

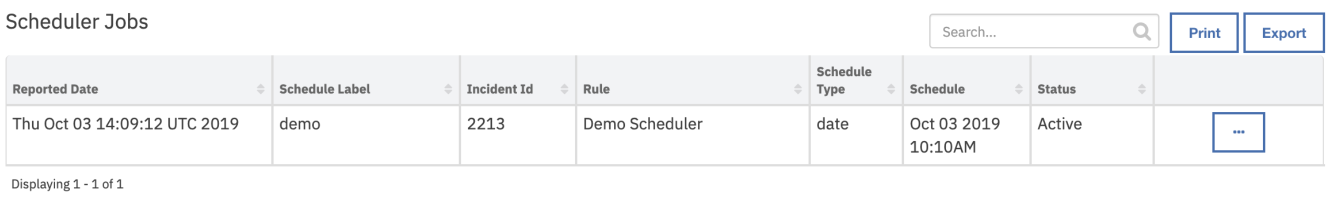
```
inputs.scheduler_type = rule.properties.schedule_type
if rule.properties.schedule_type == 'date':
  # date format converted to use dashes
  inputs.scheduler_type_value =
rule.properties.schedule_type_value.replace("/", "-")
else:
  inputs.scheduler_type_value = rule.properties.schedule_type_value
inputs.scheduler_rule_name = rule.properties.schedule_rule_name
inputs.scheduler_rule_parameters =
rule.properties.schedule_rule_parameters
inputs.scheduler_label_prefix = rule.properties.scheduler_label_prefix
inputs.incident_id = incident.id
```

▶ Example Post-Process Script:

```
None
```

Function - List Scheduled Rules

List the schedules presently defined



Scheduler Jobs

Reported Date	Schedule Label	Incident Id	Rule	Schedule Type	Schedule	Status	
Thu Oct 03 14:09:12 UTC 2019	demo	2213	Demo Scheduler	date	Oct 03 2019 10:10AM	Active	...

Displaying 1 - 1 of 1

▶ Inputs:

Name	Type	Required	Example	Tooltip
`incident_id`	`number`	Yes	`-`	Incident Id to limit returned schedules. 0 or None return all

▶ Outputs:

```
results = {
    'success': True
    'content': [
    {
      'args': (2219, # incident_id
      None, # object_id
      None, # row_id
      u'rule3', # scheduled rule
      u'Delete rule3', # schedule rule label
      49, # rule_id
      0, # object_type_id
      None,
      None),
      'type': 'date', # schedule rule type
      'id': u'rule3', # schedule rule label
      'value': 'Oct 03 2019 12:35PM' # Schedule
    }
  ],
}
```

▶ Example Pre-Process Script:

```
if rule.properties.incidents_returned == "All":
  inputs.incident_id = 0
else:
  inputs.incident_id = incident.id
```

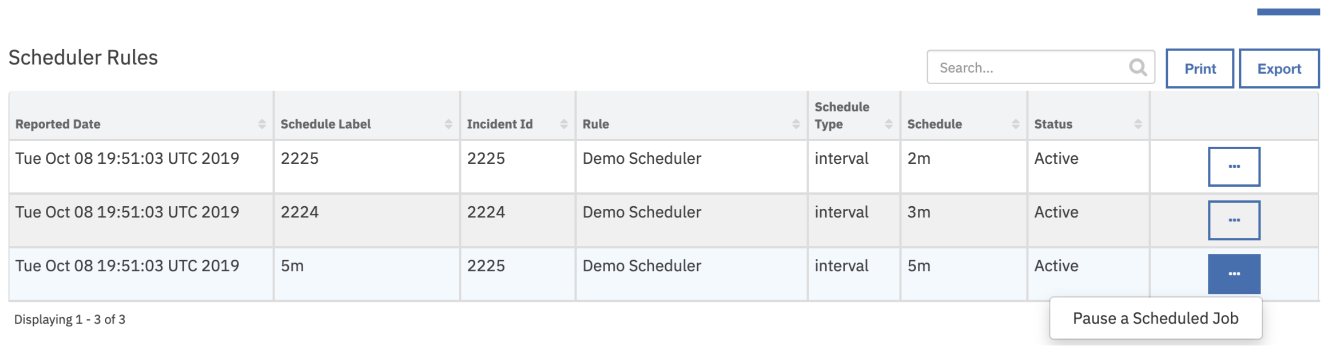
▶ Example Post-Process Script:

```
import java.util.Date as Date

if not results['content']:
  row = incident.addRow("scheduler_rules")
  row['reported_on'] = str(Date())
  row['schedule_label'] = "-- no scheduled rules --"
else:
  for job in results['content']:
    row = incident.addRow("scheduler_rules")
    row['schedule_label'] = job['id']
    row['schedule_type'] = job['type']
    row['incident_id'] = job['args'][0]
    row['rule'] = job['args'][4]
    row['schedule'] = job['value']
    row['reported_on'] = str(Date())
    row['status'] = 'Active'
```

Function - Pause a Scheduled Rule

Pause an existing scheduled rule



Reported Date	Schedule Label	Incident Id	Rule	Schedule Type	Schedule	Status	
Tue Oct 08 19:51:03 UTC 2019	2225	2225	Demo Scheduler	interval	2m	Active	...
Tue Oct 08 19:51:03 UTC 2019	2224	2224	Demo Scheduler	interval	3m	Active	...
Tue Oct 08 19:51:03 UTC 2019	5m	2225	Demo Scheduler	interval	5m	Active	...

Displaying 1 - 3 of 3

▶ Inputs:

Name	Type	Required	Example	Tooltip
`scheduler_label`	`string`	Yes	`-`	Label of scheduled job to pause

▶ Outputs:

```
results = {
  'inputs': {
    u'scheduler_label': u'2225'
  },
  'metrics': {
    'package': 'fn-scheduler',
    'timestamp': '2019-10-08 15:38:04',
    'package_version': '1.0.0',
    'host': 'marks-mbp.cambridge.ibm.com',
    'version': '1.0',
    'execution_time_ms': 21
  },
  'success': True,
  'content': {
    'args': (2225,
    None,
    None,
    u'2225',
    u'Demo Scheduler',
    39,
    0,
    {
      u'scheduler_demo': u'yes'
    },
    None),
    'type': 'interval',
    'id': u'2225',
    'value': '2m'
  },
  'raw': '{"args": [2225, null, null, "2225", "Demo Scheduler", 39, 0,
{"scheduler_demo": "yes"}, null], "type": "interval", "id": "2225",
"value": "2m"}',
  'reason': None,
  'version': '1.0'
}
```

▶ Example Pre-Process Script:

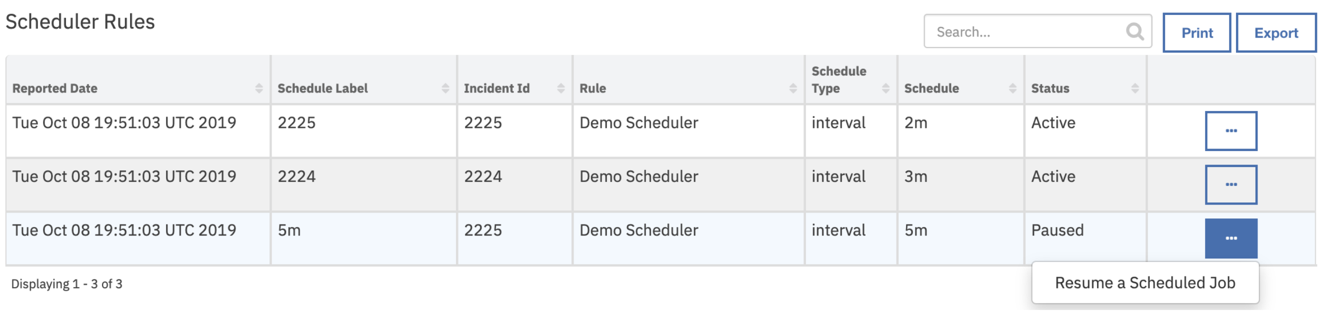
```
inputs.scheduler_label = row.schedule_label
```

▶ Example Post-Process Script:

```
if results.success:
  row['status'] = 'Paused'
else:
  row['status'] = row['status'] + " (Error)"
```

Function - Resume a Scheduled Rule

Resume an existing scheduled rule



Scheduler Rules

Reported Date	Schedule Label	Incident Id	Rule	Schedule Type	Schedule	Status	
Tue Oct 08 19:51:03 UTC 2019	2225	2225	Demo Scheduler	interval	2m	Active	...
Tue Oct 08 19:51:03 UTC 2019	2224	2224	Demo Scheduler	interval	3m	Active	...
Tue Oct 08 19:51:03 UTC 2019	5m	2225	Demo Scheduler	interval	5m	Paused	...

Displaying 1 - 3 of 3

▶ Inputs:

Name	Type	Required	Example	Tooltip
`scheduler_label`	`string`	Yes	`-`	Label of scheduled job to resume

▶ Outputs:

```
results = {
  'inputs': {
    u'scheduler_label': u'2225'
  },
  'metrics': {
    'package': 'fn-scheduler',
    'timestamp': '2019-10-08 15:38:04',
    'package_version': '1.0.0',
    'host': 'marks-mbp.cambridge.ibm.com',
    'version': '1.0',
    'execution_time_ms': 21
  },
  'success': True,
  'content': {
    'args': (2225,
    None,
    None,
    u'2225',
    u'Demo Scheduler',
    39,
    0,
    {
      u'scheduler_demo': u'yes'
    },
    None),
    'type': 'interval',
    'id': u'2225',
    'value': '2m'
  },
  'raw': '{"args": [2225, null, null, "2225", "Demo Scheduler", 39, 0,
{"scheduler_demo": "yes"}, null], "type": "interval", "id": "2225",
"value": "2m"}',
  'reason': None,
  'version': '1.0'
}
```

▶ Example Pre-Process Script:

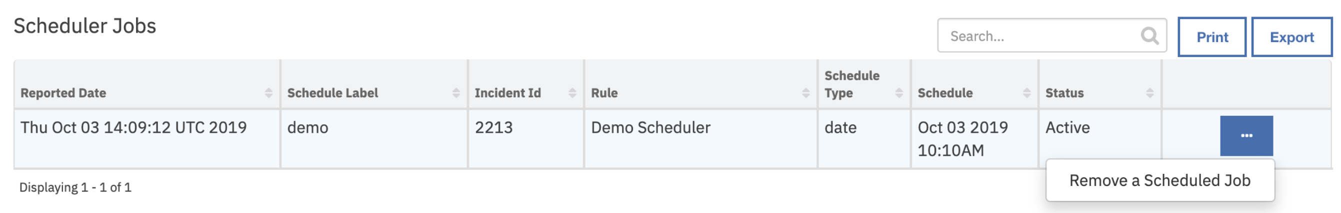
```
inputs.scheduler_label = row.schedule_label
```

▶ Example Post-Process Script:

```
if results.success:
  row['status'] = 'Active'
else:
  row['status'] = row['status'] + " (Error)"
```

Function - Remove a Scheduled Rule

Stop a schedule


Scheduler Jobs

Reported Date	Schedule Label	Incident Id	Rule	Schedule Type	Schedule	Status	
Thu Oct 03 14:09:12 UTC 2019	demo	2213	Demo Scheduler	date	Oct 03 2019 10:10AM	Active	···

Displaying 1 - 1 of 1

▶ Inputs:

Name	Type	Required	Example	Tooltip
`scheduler_label`	`text`	Yes	`-`	Label to reference created schedule

▶ Outputs:

```
results = {
    'success': True
    'content':  None
}
```

▶ Example Pre-Process Script:

```
inputs.scheduler_label = row.schedule_label
```

▶ Example Post-Process Script:

```
if results.success:
  row['status'] = "Deleted"
else:
  row['status'] = row['status'] + " (Error)"
```

Rules

Rule Name	**Object**	**Workflow Triggered**
List Scheduled Rules	incident	`list_schedules`
Remove a Scheduled Rule	scheduler_rules	`remove_a_schedule`
Schedule a Rule to Run	incident	`schedule_rule_to_run`
Schedule a Rule to Run - Artifact	artifact	`schedule_a_rule_to_run_artifact`
Schedule a Rule to Run - Task	task	`schedule_a_rule_to_run__task`

Considerations

Rules

- Rules must be enabled to be scheduled and are again checked when the scheduled rule is triggered.
- Rules scheduled must match the invoking Rule. For instance, to create a scheduled artifact rule, use the rule `Create a Schedule - Artifact`.
- All schedules must be in the future.
- Disabled rules will not execute but the scheduled rule will continue to trigger.
- Rules triggered on closed incidents will not run and the scheduled rule will be removed.
- Incident notes are created each time a scheduled rule is executed documenting the rule invocation.
- Scheduled rules will not show up under Action Status and Workflow Status. Refer instead to the incident notes.

Artifacts

- Rules executed against artifacts should include at least two Activity Fields:
  - artifact_type
  - artifact_value
- Your artifact level workflow and function would then capture this information using rule properties such as:
  - inputs.artifact_type = rule.properties.artifact_type
  - inputs.artifact_value = rule.properties.artifact_value

Datatables

- Datatable scheduled rules are not part of this package, but can be easily created for a specific Datatable.
- Datatable scheduled rules cannot currently reference the invoking datatable row in the pre-processing script. However, a rule's activity field can be defined to prompt for it.

Persistence of Scheduled Rules

- Labels for scheduled rules need to be unique. Attempting to create a duplicate scheduled rule label will fail.
- Sqlite is used to persist scheduled rules. Restarting resilient-circuits will continue with the scheduled rules already defined.

Integrations

- A function executed from a scheduled rule is free to perform any operation against Resilient. Even through a scheduled rule runs from a specific Incident, Resilient API calls can collect and operate on other incidents. For example, a scheduled rule can call a function which queries Resilient for all open tasks with due dates to review any overdue.