API

This section contains the following topics:

CWM API Overview

The CWM API was developed according to the Representational State Transfer (REST) design principles. The API is accessed using HTTP with JSON data format. The success or failure of the request is indicated by the relevant HTTP response code. Data retrieval methods require a GET request, while methods for adding, changing, or deleting data require POST, PUT, PATCH, or DELETE methods. Errors will be returned if the request is sent with the wrong request type.

How to use CWM API?

You can consume CWM API in two ways:

- via Swagger interface, or
- via Postman collection.

Built directly into the product is a Swagger interface accessed from the CWM UI, but for ease of use, a Postman collection with example requests is also provided.

All tutorials under *Manage via API* section assume the use of Swagger.

Use Swagger

To access CWM Swagger API, from the navigation menu on the left, click the **swagger** icon.

Use CWM Postman collection

Prerequisites

- Postman Web app account or Postman Desktop installed.

Download JSON collection file

Download the Postman collection in JSON format by clicking this link. Unpack the zip archive.

Import collection and set environment

Step 1 Open Postman and go to **Collections**.

Step 2 Click **Import**, select **folders** from the **Drop anywhere to import** screen and point to the folder that you have unpacked from the zip archive.

Step 3 Go to **Environments** and select the newly imported **test** environment.

Step 4 Provide current values for the **baseUrl** and **port** variables to fit your CWM IP address and port and save the changes.

Now you're set up and ready to use the collection.

Manage adapters

To interact with external target systems, CWM requires adapters. You can manage them in the CWM UI as described in the **Operator** guide, or using the CWM API. The following API endpoints are available for handling adapters:

- `GET/adapter`: gets a list of adapters existing in the CWM application.
- `POST/adapter`: uploads an adapter **.tar** file to CWM storage.
- `GET/adapter/{adapterId}`: gets the details of a specific adapter existing in the CWM application. Among others, it lists all the activities available in the adapter.
- `PUT/adapter/{adapterId}`: updates an existing adapter file with a new adapter version.
- `DELETE/adapter/{adapterId}`: deletes an adapter from the CWM application.
- `POST/adapter/{adapterId}/deploy`: deploys an adapter in the system based on the uploaded adapter file.
- `PATCH/adapter/{adapterId}`: updates the default status of the adapter.

Install adapter

CWM adapters come in **.tar** installation files. Before they can be used in a workflow, they need to be uploaded to storage and deployed in the system. Here's how to do it.

Upload adapter file

Before you deploy an adapter, you need to upload the adapter **.tar** file to CWM storage:

Step 1 Get a latest adapter installation file or create your own adapter.

Step 2 Log in to CWM and from the navigation menu on the left, click the :simple-swagger: icon.

Step 3 In the **adapters** section, click the `POST/adapter` endpoint to expand it. Inside the endpoint, click **Try it out**.

Step 4 In the subsection that appears, click **Choose File**, select the adapter **.tar** installation file and click **Upload**, then click **Execute**.

If the server response code is `201`, the adapter file is successfully uploaded into the CWM database.

Deploy adapter

Step 1 In the CWM API **adapters** section, click the `GET/adapter` endpoint to expand it. Inside the endpoint, click **Try it out** and **Execute**.

Step 2 From the server response body, copy the value of the `id` field for your uploaded adapter.

Step 3 In the CWM API **adapters** section, click the `POST/adapter/{adapterId}/deploy` endpoint to expand it.

Step 4 Inside the endpoint, click **Try it out**. Paste the adapter id into the **Adapter ID** field.

Step 5 In the **createWorker** field, you may set the `createWorker` parameter to `true`. This will create a worker with the same name as the adapter id.

Step 6 Click **Execute**.

If the server response code is `201`, the adapter plugin is successfully installed and you're good to proceed.

Delete adapter

To delete an adapter permanently from storage and uninstall it:

Step 1 In the CWM API **adapters** section, click the `GET/adapter` endpoint to expand it. Inside the endpoint, click **Try it out** and **Execute**.

Step 2 From the server response body, copy the value of the `id` field for your uploaded adapter.

Step 3 In the CWM API **adapters** section, click the `DELETE/adapter/{adapterId}` endpoint to expand it.

Step 4 Inside the endpoint, click **Try it out**. Paste the adapter id into the **Adapter ID** field.

Step 5 Click **Execute**.

Manage workers

Workers are processes that execute actions defined in workflow definitions and adapter code. You can manage them using the CWM UI as described in the **Operator** guide, or with the CWM API, as described below.

The following actions for managing workers are available:

- `GET/worker`: gets a list of workers existing in the CWM application.
- `POST/worker`: creates a new worker in the CWM application.
- `GET/worker/{workerName}`: gets the details of a specific worker existing in the CWM application.
- `PUT/worker/{workerName}`: updates an existing worker with new parameter values.
- `DELETE/worker/{workerName}`: deletes a worker from the CWM application.
- `POST/worker/{workerName}/start`: activates a worker created in the application.
- `POST/worker/{workerName}/stop`: deactivates a worker created in the application.

Create worker

Step 1 Log in to CWM and from the navigation menu on the left, click the **swagger** icon.

Step 2 In the CWM API **workers** section, click the `POST/worker` endpoint to expand it. Inside the endpoint, click **Try it out**.

Step 3 In the **Worker data** field, provide the required values:

a) "activities": paste the ID of your deployed adapter or specific adapter activity.
b) "startWorker": set to `true`.
c) "workerName": provide a name for your worker.

Step 4 Click **Execute**.

Start a worker

Step 1 In the CWM API **workers** section, click the `POST/{workerName}/start` endpoint to expand it. Inside the endpoint, click **Try it out**.

Step 2 In the **parameters** fields, provide the required values:

a) "Name of a worker to start": paste the name the worker to be started.
b) "forceReload": set to `true` if you want to force the worker to start.

Step 3 Click **Execute**.

Stop a worker

Step 1 In the CWM API **workers** section, click the `POST/{workerName}/stop` endpoint to expand it. Inside the endpoint, click **Try it out**.

Step 2 In the **parameters** fields, provide the required values:

a) "Name of a worker to stop": paste the name the worker to be stopped.

b) "forceStop": set to `true` if you want to force the worker to stop.

Step 3 Click **Execute**.

Manage workflows

Workflow definitions can be managed both in the CWM UI as described in the **Operator** guide, or using the CWM API:

- `GET/workflow`: gets a list of workflow definitions existing in the CWM application.
- `POST/workflow`: creates a new workflow definition in the CWM application.
- `GET/workflow/{workflowId}`: gets the details of a specific workflow existing in the CWM application.
- `PUT/workflow/{workflowId}`: updates an existing workflow definition in the CWM application.
- `DELETE/workflow/{workflowId}`: deletes a selected workflow definition from the CWM application.
- `GET/workflowExport`: exports workflow definitions based on a given array of workflow definition IDs.
- `POST/workflowImport`: imports in bulk workflow definitions to the CWM application.

Note The recommended method for managing workflows is through CWM UI. For details, refer to the **Operator** guide.

Manage resources and secrets

Overview

For CWM, adapters define activities that enable to execute actions in external entities, such as other systems or applications. These entities are, in most cases, integrated via APIs which usually require connection and authentication data. CWM provides a framework where when an activity is consumed in a workflow, the details of a connection endpoint and authentication data can be passed at runtime. Thus, the operator who runs a workflow may not know any details of these systems (resources) such as IP addresses, ports or usernames and passwords.

CWM provides a framework for secure handling of resources and secrets in the database and identifying them by their respective IDs. When running a workflow, just the resource ID needs to be passed, with the rest of the data sent to the adapter by the Resource Manager without any intervention from the Operator or additional development from Adapter developer. You can manage secrets and resources in the CWM UI as described in the **Operator** guide, or using the CWM API.

Resource and secret types

You can think of resource and secret types are buckets used to organize resources and secrets created by users by their type. Types are defined inside a given adapter and are added to the system automatically upon installing the adapter. You can list secrets belonging to a specific type using the `GET/secretType/{secretTypeId}` API endpoint.

Secrets API endpoints

The following actions for managing secrets are available:

- `GET/secret`: gets a list of secrets existing in the CWM application.
- `POST/secret`: creates a new secret in the CWM application.
- `GET/secretType/{secretTypeId}`: lists secrets existing in the CWM application that belong to a specific type.
- `GET/secretType`: gets a list of secret types existing in the CWM application.
- `GET/secret/{secretId}`: gets details of an existing secret.
- `DELETE/secret/{secretId}`: deletes a secret from the CWM application.
- `PATCH/secret/{secretId}`: updates a secret existing in the CWM application with new parameter values.

Resources API endpoints

The following actions for managing resources are available:

- `GET/resource`: gets a list of resources existing in the CWM application.
- `POST/resource`: creates a new resource in the CWM application.
- `GET/resource/{resourceId}`: gets the details of a specific resource existing in the CWM application.
- `PATCH/resource/{resourceId}`: updates an existing resource with new parameter values.
- `DELETE/resource/{resourceId}`: deletes a resource from the CWM application.
- `GET/resourceType`: gets a list of resource types existing in the CWM application.
- `GET/resourceType/{resourceTypeId}`: gets the details of an existing resource type.

Create secret

Step 1 Log in to CWM and from the navigation menu on the left, click the :simple-swagger: icon.

Step 2 In the CWM API **secrets** section, click the `POST /secret` endpoint to expand it.

Step 3 Inside the endpoint, click **Try it out**, and provide your data into the **Secret input** field. Example input can look like this:

```
{
"secret": {
"username": "admin",
"password": "admin"
},
"secretId": "NSOSecret",
"secretType": "basicAuth"
}
```

Step 4 Click **Execute**.

If the server response code is `201`, the secret is successfully created and you can start creating a resource to associate the secret with.

Create resource

SUMMARY STEPS

1. In the CWM API **resources** section, click the `POST /resource` endpoint to expand it.
2. Inside the endpoint, click **Try it out**, and provide your data into the **Resource input** field. Example input can look like this:
3. Click **Execute**.

DETAILED STEPS

	Command or Action	Purpose
Step 1	In the CWM API **resources** section, click the `POST /resource` endpoint to expand it.	
Step 2	Inside the endpoint, click **Try it out**, and provide your data into the **Resource input** field. Example input can look like this:	`{ "resource": { "scheme": "http", "host": "127.0.0.1", "port": 8080 }, "resourceId": "NSOLocal", "resourceType": "cisco.nso.resource.v1.0.0", "secretId": "NSOSecret" }`
Step 3	Click **Execute**.	If the server response code is `201`, the resource is successfully created.

Manage Schedules

To automate recurring operations or settle them on a specific date and time in the future, you can create a schedule. In CWM, there are two types of schedules:

- One-time: define at what time and date a single job will be executed

- Recurring: define rules (based on the interval, calendar or cron expression) when the job run repeats

CWM scheduler API allows you to create, update and pause/unpause schedules. While creating a basic schedule (along with other operations like deleting a schedule) in {{ version.CWM }} is possible via UI, defining advanced schedules using more functionalities of the scheduler is available only via API.

The following API endpoints are available for handling schedules:

- `GET/schedule`: gets a list of schedules existing in the CWM application.
- `POST/schedule`: creates a new schedule in the CWM application.
- `GET/schedule/{scheduleId}`: gets the details of a specific schedule existing in the CWM application.
- `PATCH/schedule/{scheduleId}`: updates an existing schedule with new detail(s).
- `DELETE/schedule/{scheduleId}`: deletes a schedule from the CWM application.

Create a schedule

To create a schedule, follow the steps below:

SUMMARY STEPS

1. Log in to CWM and from the navigation menu on the left, click the :simple-swagger: icon.
2. In the CWM API **scheduler** section, click the `POST/schedule` endpoint to expand it.
3. Inside the endpoint, click **Try it out** and in the **Schedule request** provide the chosen values.
4. Click **Execute**. If the server response code is `201`, the schedule has been successfully created.

DETAILED STEPS

	Command or Action	Purpose
Step 1	Log in to CWM and from the navigation menu on the left, click the :simple-swagger: icon.	
Step 2	In the CWM API **scheduler** section, click the `POST/schedule` endpoint to expand it.	
Step 3	Inside the endpoint, click **Try it out** and in the **Schedule request** provide the chosen values.	
Step 4	Click **Execute**. If the server response code is `201`, the schedule has been successfully created.	

Update a schedule

To edit a schedule, follow the steps below:

Step 1 Log in to CWM and from the navigation menu on the left, click the :simple-swagger: icon.

Step 2 In the CWM API **scheduler** section, click the `PATCH/schedule` endpoint to expand it.

Step 3 Inside the endpoint, click **Try it out**, provide the *Schedule ID* and in the **Schedule update request** edit the chosen values.

Step 4 Click **Execute**. If the server response code is `200`, the schedule has been successfully updated.

Updating a schedule replaces the entire configuration. It means that if you want to change only one existing value, you still need to pass all of the details again, even if they will be the same.

Pause a schedule

You can pause a schedule for a chosen time, for example, for a maintenance window. When the schedule is paused, the runs that are supposed to be executed are skipped. To pause a schedule, follow the steps below:

Step 1 Log in to CWM and from the navigation menu on the left, click the :simple-swagger: icon.

Step 2 In the CWM API **scheduler** section, click the `PATCH/schedule` endpoint to expand it.

Step 3 Inside the endpoint, click **Try it out**, provide the *Schedule ID* and in the **Schedule update request** set the value of the field "paused" to `true`.

Step 4 Click **Execute**. If the server response code is `200`, the schedule has been successfully paused.

The schedule remains paused as long as you resume it with the next request.

Unpause a schedule

To resume a schedule, follow the steps below:

Step 1 Log in to CWM and from the navigation menu on the left, click the swagger icon.

Step 2 In the CWM API **scheduler** section, click the `PATCH/schedule` endpoint to expand it.

Step 3 Inside the endpoint, click **Try it out**, provide the *Schedule ID* and in the **Schedule update request** set the value of the field "paused" to `false`.

Step 4 Click **Execute**. If the server response code is `200`, the schedule has been successfully resumed.

Pause on failure

If you set `pauseOnFailure` field to `true`, the schedule will be automatically paused after any of its job fails. It gives a chance to address the issue, for example, when the workflow definition associated with the schedule will be deleted. To change the pause on failure value, follow the general steps in **Update a schedule**.

Change the overlap policy

Overlap policy controls what happens when the next job should be started by a schedule at the same time that a previous run is still being executed. The default policy is **Skip** (with the "overlap" field value set to `1`), which means that when the previous job is still running, the next scheduled run won't start and it will be skipped. When the existing run completes, only the next scheduled run after that time will be considered. To change the overlap policy, follow the general steps in **Update a schedule**.

Possible policies:

Policy type	`overlap` field value	Description
Skip	1	This is the default policy that prevents overlapping runs. While the previous run from a schedule is still running when the next one should be executed, the next run will be skipped.
Buffer One	2	Starts the next run as soon as the current one completes. Only one run will be buffered. If they are more runs that are supposed to happen when the current job is running, they are skipped, and only the first one in the queue will be executed after the running job finishes.
Buffer All	3	Buffers all runs that are supposed to happen when the current job is running. All buffered runs will be executed sequentially, directly after the running job completes.
Cancel Other	4	Cancels the running job and starts the new one after the cancellation of the old one is completed.
Terminate Other	5	Terminates running job and starts the new one immediately.
Allow All	6	Starts any number of concurrent runs.

Manage Jobs

Jobs can be managed both in the CWM UI as described in the Operator guide or using the CWM API.

!!! note "Important" Some functionalities, for example, querying the jobs based on multiple tags, for {{ version.CWM }} are available only via API.

The following actions for managing jobs are available:

- `GET/job`: gets a list of jobs existing in the CWM application.
- `POST/job`: creates a new job run in the CWM application based on given parameters.
- `GET/job/{jobId}/runs/{runId}`: returns the details of a specific job existing in the CWM application.
- `POST/job/{jobId}/runs/{runId}/cancel`: cancels the execution of a running job, after the workflow worker completes the ongoing task execution from the workflow definition.
- `GET/job/{jobId}/runs/{runId}/events`: returns the event history for a specific job.
- `POST/job/{jobId}/runs/{runId}/terminate`: immediately terminates the execution of a running job.

Execute a job

To run a job, follow the steps below:

Step 1 Log in to CWM and from the navigation menu on the left, click the :simple-swagger: icon.

Step 2 In the CWM API **jobs** section, click the `POST/job` endpoint to expand it.

Step 3 Inside the endpoint, click **Try it out** and in the **Job Execution Request** provide the chosen values, for example:

```
{
"data": {},
"jobName": "test API job",
"tags": [
    "test", "API"
],
"workflowName": "Test cisco workflow",
"workflowVersion": "1.1"
}
```

Note Job tags are optional but may ease filtering specific jobs in the future.

Step 4 Click **Execute**.

If the job run has been successfully created, you should get a server response with a code `200` and the job ID and run ID returned.

Filter jobs by multiple tags

You can get a list of jobs existing in CWM filtered by specific queries, for example, by associated tags. To get a list, follow the steps below:

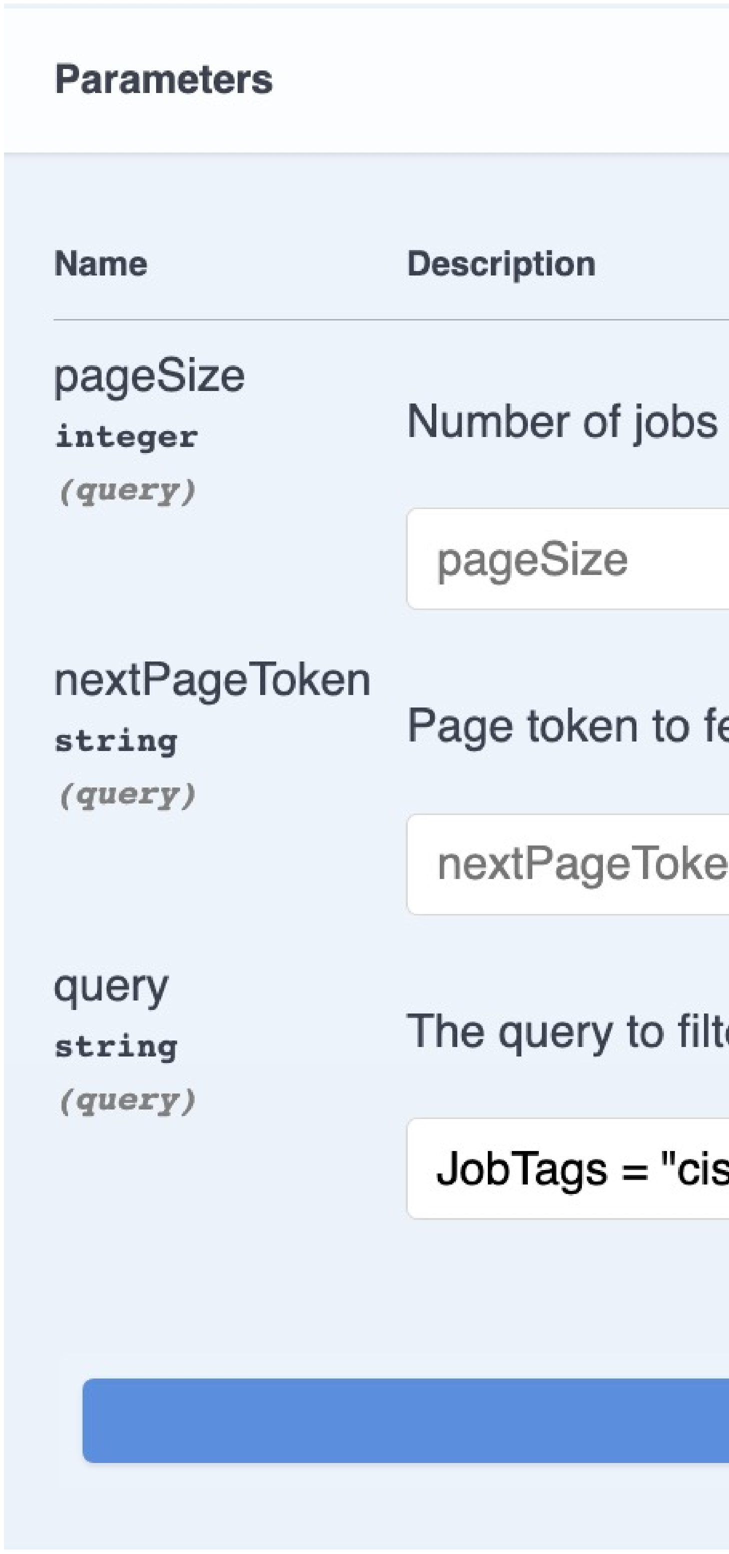
SUMMARY STEPS

1. Log in to CWM and from the navigation menu on the left, click the :simple-swagger: icon.
2. In the CWM API **jobs** section, click the `GET/job/{jobId}/runs/{runId}` endpoint to expand it.
3. Inside the endpoint, click **Try it out** and in the `query` parameter, specify the tags by which you want to filter the jobs, following the schema: `JobTags = "tag_name1" and JobTags = "tag_name2"`, as in the example below:
4. Click **Execute**.

DETAILED STEPS

	Command or Action	Purpose
Step 1	Log in to CWM and from the navigation menu on the left, click the :simple-swagger: icon.	
Step 2	In the CWM API **jobs** section, click the `GET/job/{jobId}/runs/{runId}` endpoint to expand it.	

	Command or Action	Purpose
Step 3	Inside the endpoint, click **Try it out** and in the `query` parameter, specify the tags by which you want to filter the jobs, following the schema: `JobTags = "tag_name1" and JobTags = "tag_name2"`, as in the example below:	***Figure 1: Job Event Log***

	Command or Action	Purpose
		

Parameters

Name	Description
pageSize integer (query)	Number of jobs
nextPageToken string (query)	Page token to fe
query string (query)	The query to filte

JobTags = "cis

	Command or Action	Purpose
Step 4	Click **Execute**.	You should receive a server response with a `200` status code, along with the filtered jobs details. **Known issue**: In the response body with the job details, the field named `workflowId` is actually a Job ID, not a Workflow definition ID.

Get event history

To get a list of event history (all or filtered) for a given job, follow the steps below:

Step 1 Log in to CWM and from the navigation menu on the left, click the **swagger** icon.

Step 2 In the CWM API **jobs** section, click the `GET/job/{jobId}/runs/{runId}/events` endpoint to expand it.

Step 3 Inside the endpoint, click **Try it out** and provide the Run ID and Job ID of the job for which you want to get the event history.

Step 4 Optionally, you can set `isLongPoll` parameter to `true` if you are querying the currently running job. Then, the connection will be open until the execution of a given job finishes, and you will get the response after the job execution is completed. If you set it to `false`, you will receive a history of events immediately after sending the request, consisting of events completed up to the moment of request.

Step 5 Optionally, you can set `filterType` parameter to the chosen value (`all` or `close_event`).

Note The `close_event` value filters only for the closing event of an already finished job, for example `WorkflowExecutionFailed` or `WorkflowExecutionCompleted`. If you picked the `close_event` as a filter, and your job is currently running, you will receive a `400` error.

Step 6 Click **Execute**.

You should receive a server response with a `200` status code, along with the filtered events.

Manage policies

The CWM policy API allows you to manage policies that define rules for user access as part of the RBAC functionality in CWM. You can define policies only through the API.

!!! note When you define a policy, at the same time you create a new CWM user role (or update an existing role). Therefore, a `roleId` (where role ID is the `role_name` from the schema `permission/role_name`, without the "permission/" prefix) is required when defining a policy. If the role already exists, the new policy will be matched to it. If you create a new role, you will need to add it to a user in CWM afterwards. For a detailed instruction, see the **Manage access policy for users** section.

The following API endpoints are available for handling policies:

- `GET/policy/{roleId}`: gets the details of a specific policy existing in CWM.
- `PUT/policy/{roleId}`: updates a specific policy existing in CWM.

- `POST/policy/{roleId}`: creates a new policy in CWM.

Create a policy

To create a policy, follow the steps below:

Step 1 Log in to CWM and from the navigation menu on the left, click the :simple-swagger: icon.

Step 2 In the CWM API **policy** section, click the `POST/policy/{roleId}` endpoint to expand it.

Step 3 Inside the endpoint, click **Try it out** and in the **roleId**, provide the CWM role name.

Step 4 Click **Execute**. If the server response code is `200`, the policy has been successfully created.

Note Note that by default, the policy has all the permissions set to `false`. To modify the permissions, you need to update the policy.

Update a policy

To modify user permissions in a policy, follow the steps below:

Step 1 In the CWM API, click the `GET/policy/{roleId}` endpoint to expand it. Inside the endpoint, click **Try it out**.

Step 2 Type the role name inside the **roleId** field and click **Execute**.

Step 3 In the API response, you'll get a JSON array. Copy the response body, edit it to reflect your needs regarding specific permissions, and copy the modified content.

Step 4 In the CWM API, click the `PUT/policy/{roleId}` API endpoint to expand it, then click **Try it out**.

Step 5 In **roleId**, type the role name.

Step 6 In **Policy to update**, paste the copied policy JSON array inside the `"policy": {}` brackets.

Step 7 Click **Execute**. Your policy will be updated. To see if it works as intended, log in as a user to which the role is assigned.

Note If the server response code is `200`, the policy has been successfully updated.

Manage Event types

Event types are categories of signals either received or produced by CWM and referred to in workflow definitions. You can manage them using the CWM UI as described in the **Operator** guide, or with the CWM API, as described below.

The following actions for managing Event types are available:

- `GET/eventType`: gets a list of Event types existing in the CWM application.
- `POST/eventType`: creates a new Event type in the CWM application.

- `GET/eventType/{name}`: gets the details of a specific Event type existing in the CWM application.
- `PUT/eventType/{name}`: updates an existing Event type with new parameter values.
- `DELETE/eventType/{name}`: deletes a Event type from the CWM application.
- `POST/eventType/{name}/start`: starts or stops an event listener.

Create Event type

Step 1 Log in to CWM and from the navigation menu on the left, click the :simple-swagger: icon.

Step 2 In the CWM API **eventType** section, click the `POST/eventType` endpoint to expand it. Inside the endpoint, click **Try it out**.

Step 3 In the **eventType data** field, modify the required values:

```
    {
"correlation": [
    {
    "contextAttributeName": "string",
    "contextAttributeValue": "string"
    }
],
"createWorkflow": false,
"dataOnly": true,
"endpoint": "string",
"kind": "string",
"name": "string",
"resourceId": "string",
"source": "string",
"type": "string",
"workflowName": "string",
"workflowVersion": "string"
}
```

Step 4 Click **Execute**.

Start/stop an Event type listener

Step 1 In the CWM API **eventType** section, click the `POST/{name}/{action}` endpoint to expand it. Inside the endpoint, click **Try it out**.

Step 2 In the **parameters** fields, provide the required values:

a) "Name": paste the name the Event type for which a listener needs to be started/stopped.

b) "action": set to `start` if you want to start a listener, or `stop` if you want to stop a running listener.

Step 3 Click **Execute**.