Cheat sheet

Red Hat Insights API

Red Hat Insights allows you to investigate and make changes to the configuration of hosts managed by Red Hat through REST APIs. This cheat sheet covers the basic APIs. All examples are performed on https://console.redhat.com/api/ using v1 endpoints. Refer to https://console.redhat.com/docs/api for the latest API specifications and deprecations.

Authentication

The Insights API provides secure REST services over HTTPS endpoints. This protects authentication credentials in transit.

Basic authentication (discontinued from Dec 2024)

Important: Starting Dec 2024, Red Hat will discontinue support for basic authorization to connect to services' APIs. The Red Hat Hybrid Cloud Console is integrating service accounts with User Access functionality to provide granular control over access permissions and enhance security. Token-based authentication is recommended. More information about transitioning from basic authentication to token-based authentication via service accounts is described on the Red Hat Customer Portal.

1. Create a base64 encoding for your username and password, submitting them with the syntax `<name>:<password>:`
   `echo -n 'admin:<password>' | openssl base64`
2. Include the `<encoded_auth>` output in an `Authorization: Basic` HTTP header in your request:
   `Authorization: Basic <encoded_auth>`

Red Hat token-based authentication via service account (recommended)

1. Create a new service account on the Red Hat Hybrid Cloud Console.
2. Obtain service account credentials (Client ID and Client Secret).
3. Add the service account to the applicable User Access Groups (service accounts do not inherit permissions from Default access groups, so they must be granted access permissions by placing them in a User Group).
4. Generate an access token using Red Hat's single sign-on (SSO) technology using the service account credentials `<client_id>` and `<client_secret>`

```
curl
https://sso.redhat.com/auth/realms/redhat-external/protocol/openid-connect/
token -d "grant_type=client_credentials" -d "scope=api.console" -d
client_id=<client_id> -d client_secret=<client_secret>
```

5. Red Hat's SSO will respond with a JSON object containing an access token. This access token can be used to make authenticated requests to APIs by including the access token generated in the previous step in an `Authorization: Bearer` HTTP header in your request: `Authorization: Bearer <access_token>`

Red Hat API token authentication (recommended)

1. Log in to the Red Hat Customer Portal with your username and password.
2. Generate an offline token using Red Hat API Tokens by following the instructions.
3. Generate an access token, submitting the offline token generated in the previous step:

```
curl
https://sso.redhat.com/auth/realms/redhat-external/protocol/openid-connect/
token -d grant_type=refresh_token -d client_id=rhsm-api -d
refresh_token=<offline_token>
```

4. Include the access token generated in the previous step in an `Authorization: Bearer` HTTP header in your request: `Authorization: Bearer <access_token>`

Common activities

Host inventory

Get all hosts in the account:

`GET /inventory/v1/hosts`

Get system details (e.g., after registration, using the UUID provided):

`GET /inventory/v1/hosts/<uuid>`

Get facts about all systems' system profiles:

`GET /inventory/v1/hosts/<uuid>/system_profile`

Get facts about a specific system's system profile (e.g., `last_boot_time` and `os_kernel_version`):

`GET /inventory/v1/hosts/<uuid>/system_profile?fields[system_profile]=last_boot_time,os_kernel_version`

> Only the following facts are syndicated at present: `bios_release_date, bios_vendor, bios_version, infrastructure_type, operating_system, owner_id, rhsm, sap_sids, sap_system.`

Get a system's tags:

`GET /inventory/v1/hosts/<uuid>/tags`

Get a subset of systems (using a filter on system profile):

`GET /inventory/v1/hosts?filter[system_profile][infrastructure_type]=virtual`

Delete a system:

`DELETE /inventory/v1/hosts/<uuid>`

Advisor

Get all active hits for the account:

`GET /insights/v1/rule/`

Get all rule hits on hosts:

`GET /insights/v1/export/hits/`

> Exports are available as CSV and JSON.

Get all active hits with Ansible remediation playbooks:

`GET /insights/v1/export/hits?has_playbook=true`

Get summary of all hits for a given system :

`GET /insights/v1/system/<uuid>`

Drift

Get defined baselines:

`GET /system-baseline/v1/baselines`

Create a new baseline by passing a JSON request body with a `baseline_facts` or a `inventory_uuid` or `hsp_uuid` to copy the baseline facts from, e.g.:

```
POST /system-baseline/v1/baselines
{
  “baseline_facts”: [
    {
      “name”: “arch”,
      “value”: “x86_64”
    }
  ],
  “display_name”: “my_baseline”
}
```

> `DELETE` and `PATCH` operations are also available on `/system-baseline/v1/baselines/<baseline_id>`.

Run a comparison, passing a list of systems, baselines, historical system profiles, and a reference for comparison (multiple UUIDs or other items are formatted as comma-separated lists):

```
GET /drift/v1/comparison_report?system_ids[]=<uuids>,baseline_ids[]=<baseline_ids>,historical_system_profile_ids[]=
<hsp_ids>,reference_id=<id>
```

Get historical system profiles on a system:

`GET /historical-system-profiles/v1/systems/<uuid>`

Get a specific historical system profile on a system:

`GET /historical-system-profiles/v1/profiles/<profile_id>`

Vulnerabilities

Get vulnerabilities affecting systems in the account:

`GET /vulnerability/v1/vulnerabilities/cves?affecting=true`

Get executive reports, e.g., CVEs by severity, top CVEs, etc.:

`GET /vulnerability/v1/report/executive`

Compliance

Get systems associated with Security Content Automation Protocol (SCAP) policies:

`GET /compliance/v1/systems`

Get systems' compliance/failures for defined reports:

`GET /compliance/v1/profiles`

Policies

Get all defined policies:

`GET /policies/v1/policies`

Create a new policy:

```
POST /policies/v1/policies
{
  “name”: “my_policy”,
  “description”: “My policy”,
  “isEnabled”: true,
  “conditions”: “arch = \”x86_64\””,
  “actions”: “notification”
}
```

> `DELETE` and `PUT` operations are also available on `/policies/<policy_id>`.

Get all systems triggering a policy:

`GET /policies/v1/policies/<policy_id>/history/trigger`

Patches

Get all systems with applicable advisories (patches available):

`GET /patch/v3/advisories`

Get all applicable advisories for a specific system:

`GET /patch/v3/systems/<uuid>/advisories`

Subscriptions

Get all subscribed Red Hat Enterprise Linux systems matching filters (e.g., Premium SLA, Production usage):

`GET /rhsm-subscriptions/v1/hosts/products/RHEL?sla=Premium&usage=Production`

Remediations

Get a list of defined remediations:

`GET /remediations/v1/remediations`

Create a new remediation and assign systems:

```
POST /remediations/v1/remediations
{
  “name”: “Fix Critical CVEs”,
  “archived”: true,
  “auto_reboot”: true,
  “add”: {
    “issues”: [
      {
        “id”: “advisor:CVE_2017_6074_kernel|KERNEL_CVE_2017_6074”,
        “resolution”: “mitigate”,
        “systems”: [
          “<uuid>”
        ]
      }
    ]
  }
}
```

> `DELETE` and `PATCH` operations are also available on `/remediations/v1/remediations/<remediation_id>`.

Get an Ansible remediation playbook:

`GET /remediations/v1/remediations/<remediation_id>/playbook`

Execute a remediation:

`POST /remediations/v1/remediations/<remediation_id>/playbook_runs`

Integrations and notifications

Get event log history for a list of last triggered Insights events and actions:

```
GET
/notifications/v1/notifications/events?endDate=2021-11-23&limit=20&offset=0&sortBy=created%3ADESC&startDate=2021-11-09
```

Get list of configured third party integrations:

`GET /integrations/v1/endpoints`

Python examples

The following Python code interacts with the Insights API using the `requests` library to abstract away the complexity of handling HTTP requests.

```
$ python -m pip install requests
```

Authentication

`>>> headers = {‘Authorization’: ‘Basic <encoded_auth>’}` or `>>> headers = {‘Authorization’: ‘Bearer <access_token>’}`

GET

```
>>> import requests
>>> insights_api_url = “https://console.redhat.com/api/inventory/v1/hosts”
>>> response = requests.get(insights_api_url, headers=headers)
>>> response.status_code
200
>>> response.json()
{‘total’: 1195, ‘count’: 50, ‘page’: 1, ‘per_page’: 50, ‘results’:
[{‘insights_id’: ‘<uuid>’, […]
```

POST

```
>>> import requests
>>> insights_api_url = “https://console.redhat.com/api/system-baseline/v1/
baselines”
```

```
>>> baseline = {"baseline_facts": [{"name": "arch", "value": "x86_64"}],
"display_name": "my_baseline"}
>>> response = requests.post(insights_api_url, headers=headers,
json=baseline)
>>> response.status_code
200
>>> response.json()
{'account': '<account_id>', 'baseline_facts': [{'name': 'arch', 'value':
'x86_64'}], 'created': '2021-11-29T21:06:33.630905Z', 'display_name':
'my_baseline', 'fact_count': 1, 'id': '<baseline_id>', 'mapped_system_count':
0, 'notifications_enabled': True, 'updated': '2021-11-29T21:06:33.630910Z'}
```

Ansible example

The following Ansible playbook uses the `ansible.builtin.uri` module to interact with the Insights API.

```
–
- hosts: localhost
  connection: local
  gather_facts: no

  vars:
    insights_api_url: "https://console.redhat.com/api"
```

```
 insights_auth: "Basic <encoded_auth>"
```

or

```
 insights_auth: "Bearer <access_token>"
```

```
 tasks:
  - name: Get Inventory
    uri:
      url: "{{ insights_api_url }}/inventory/v1/hosts/"
      method: GET
      return_content: yes
      headers:
         Authorization: "{{ insights_auth }}"
      status_code: 200
   register: result

  - name: Display inventory
    debug:
      var: result.json
```