CDP Public Cloud 7

Ranger API

Date published: 2022-08-22
Date modified: 2023-01-06

https://docs.cloudera.com/

Legal Notice

Contents

Ranger API Overview

The Ranger REST API provides a framework to enable, monitor and manage comprehensive data security across the CDP platform.

Apache Ranger currently provides a centralized security administration, fine grain access control and detailed auditing for user access within Apache Hadoop, Apache Hive, Apache HBase and other Apache components. The Ranger REST API documentation is served on the same host and port as the Ranger Admin Web UI, and does not require an extra process or extra configuration. The API supports HTTP Basic Authentication, accepting the same users and credentials as the Ranger Admin Web UI. For more information about REST APIs available for collecting Ranger metrics, see:

Related Information

Apache Ranger REST API

Ranger Admin Metrics API

You can access two APIs that fetch Ranger Admin metrics.

One Ranger Admin API returns a response in JSON format. The other Ranger Admin API returns a response in prometheus compatible format. Neither API that fetches Ranger Admin metrics requires authentication as it does not provide any sensitive information. Both APIs return the same value. You can reduce (or increase) the metrics collection interval using the hadoop-metrics2.properties file.

JSON compatible API

accessible at a url with format:

http(s)://<ADMIN_HOST>:<ADMIN_PORT>/service/metrics/json

Example: https://kg-rzmtr-1.kg-rzmtr.root.hwx.site:6182/service/metrics/json

Prometheus compatible API:

accessible at a url with format:

http(s)://<RAZ_HOST>:<RAZ_PORT>/service/metrics/prometheus

Example: https://kg-rzmtr-1.kg-rzmtr.root.hwx.site:6082/api/metrics/prometheus

Response

The response is grouped into policy, context enrichers, service, deny conditions, user group & Ranger JVM categories. Since both APIs return the same value, only json responses are discussed in this document.

Policy:

The policy group metrics gives all metrics related to policies. Further it is divided into following sub-categories.

Sample json response follows:

```
"Policy": {
        "RowFilteringCount": 0,
        "MaskingCount": 0,
        "TagAccessCount": 1,
        "ResourceAccessCount": 95,
        "ResourceAccessCountHIVE": 10,
        "ResourceAccessCountKAFKA": 12,
```

```
        "ResourceAccessCountATLAS": 9,
        "ResourceAccessCountKUDU": 3,
        "ResourceAccessCountHDFS": 3,
        ….
    }
```

Policy metric response can further be divided into row filtering, masking, resource access & tag policies.

Row Filtering: This group of metrics indicates row filtering policy metrics. Currently we only support total row filtering count metrics.

Masking: This group of metrics indicates masking policy metrics. Currently we only support total masking count metrics.

Resource Access: This group of metrics indicates resource access policy metrics. Other than total resource access policy count, it also gives component specific resource access policy count.

Sample json follows:

```
"Policy": {
        ….
        "ResourceAccessCount": 95,
        "ResourceAccessCountHIVE": 10,
        "ResourceAccessCountKAFKA": 12,
        "ResourceAccessCountATLAS": 9,
        "ResourceAccessCountKUDU": 3,
        "ResourceAccessCountHDFS": 3,
        ….
    }
```

ContextEnrichers:

This metrics group represents the available metrics for context enrichers. Similar to the d deny conditions & policy metrics group, this metric also shares count for each service definition/ type available other than total count.

Sample json follows:

```
 "ContextEnrichers": {
        "ContextEnricherCountTAG": 1,
        "ContextEnricherCount": 1
    }
```

Service:

This metrics group represents the available metrics for service. Similar to the context enrichers & policy metrics group, this metric also shares count for each service definition/ type available other than total count.

Sample json follows:

```
"Service": {
        "ServiceCount": 16,
        "ServiceCountHIVE": 1,
        "ServiceCountTAG": 1,
        "ServiceCountHBASE": 1,
        "ServiceCountADLS": 1,
        "ServiceCountKAFKA": 1,
        "ServiceCountS3": 1,
        ….
    }
```

DenyConditions:

This metrics group represents the available metrics for all deny conditions. Similar to the service & policy metrics group, this metric also shares count for each service definition/ type available other than total count.

Sample json follows:

```
"DenyConditions": {
        "DenyConditionCountTAG": 1,
        "DenyConditionCount": 1
    }
```

User Group:

This metrics group represents the available metrics for all users & groups. Other than total user & groups it also includes role specific user counts.

Sample json follows:

```
"UserGroup": {
        "UserCount": 62,
        "GroupCount": 97,
        "UserCountKeyAdmin": 1,
        "UserCountKeyAdminAuditor": 0,
        "UserCountSysAdmin": 8,
        "UserCountAdminAuditor": 0,
        "UserCountUser": 53
    }
```

Ranger JVM:

This metrics group represents the JVM metrics needed to get internal state of admin.

Sample json follows:

```
 "RangerJvm": {
        "GcTimeTotal": 5315,
        "SystemLoadAvg": 3.91,
        "ThreadsBusy": 2250,
        "GcCountTotal": 472,
        "MemoryMax": 1066401792,
        "MemoryCurrent": 179763416,
        "ThreadsWaiting": 2248,
        "ProcessorsAvailable": 4,
        "GcTimeMax": 5002,
        "ThreadsBlocked": 25,
        "ThreadsRemaining": 4
    }
}
```

Table 1: Ranger JVM Admin Metrics Descriptions

Metric	Descriptions
GcTimeTotal	Total time taken by JVM Garbage Collector.
GcTimeMax	Maximum time taken by a single JVM Garbage Collection process.
GcCountTotal	Number of times JVM Garbage Collector runs.
SystemLoadAvg	Average load on the host on which the service is running.
ThreadsBusy	Total JVM threads which are busy.

Metric	Descriptions
ThreadsWaiting	Total JVM threads which are waiting.
ThreadsBlocked	Total JVM threads which are blocked.
ThreadsRemaining	Total remaining (not busy/ waiting/ blocked) JVM threads.
MemoryMax	Maximum memory (RAM) JVM can get from the host.
MemoryCurrent	Currently occupied memory (RAM) by JVM.
ProcessorsAvailable	Currently available processor cores on host.

Ranger RAZ Metrics API

You can access two APIs that fetch Ranger RAZ metrics.

One Ranger RAZ API returns a response in JSON format. The other Ranger RAZ API returns a response in prometheus compatible format. Neither API that fetches Ranger RAZ metrics requires authentication as it does not provide any sensitive information. Both APIs return the same value. You can reduce (or increase) the metrics collection interval using the hadoop-metrics2.properties file.

JSON compatible API:

accessible at a url with format:

http(s)://<RAZ_HOST>:<RAZ_PORT>/service/metrics/json

https://kg-rzmtr-1.kg-rzmtr.root.hwx.site:6082/api/metrics/json

Prometheus compatible API:

accessible at a url with format:

http(s)://<RAZ_HOST>:<RAZ_PORT>/service/metrics/prometheus

https://kg-rzmtr-1.kg-rzmtr.root.hwx.site:6082/api/metrics/prometheus

Response

The response is grouped into raz & Ranger JVM categories. Since both APIs return the same value, only json responses are discussed in this document.

Raz:

This group of metrics gives all metrics related to RAZ application.

Sample json response follows:

```
"Raz": {
     "RequestPoolCount": 1600,
      "RequestPoolCountBUSY": 882,
     "SpoolDirSizeSOLR": 0,
     "SpoolDirSizeHDFS": 67123185,
     "AuthzTime": 109873,
     "AuthzCount": 190,
     "AuthzALLOWED": 100,
     "AuthzDENIED": 80,
     "AuthzNOTDETERMINED": 10,
```

Table 2: Raz Metrics Descriptions

Metric	Description
RequestPoolCount	Count of threads available in tomcat pool.
RequestPoolCountBUSY	Count of threads busy in tomcat pool.
SpoolDirSizeSOLR	Current size of solr spool directory.
SpoolDirSizeHDFS	Current size of HDFS spool directory.
AuthzTime	Total time taken to process all requests.
AuthzCount	Count of requests overall.
AuthzALLOWED	Count of requests allowed.
AuthzDENIED	Count of requests denied.
AuthzNOTDETERMINED	Count of requests with status not determined.

Ranger JVM:

This metrics group represents the JVM metrics needed to get internal state of admin.

Sample json response follows:

```
"RangerJvm": {
        "GcTimeTotal": 5315,
        "SystemLoadAvg": 3.91,
        "ThreadsBusy": 2250,
        "GcCountTotal": 472,
        "MemoryMax": 1066401792,
        "MemoryCurrent": 179763416,
        "ThreadsWaiting": 2248,
        "ProcessorsAvailable": 4,
        "GcTimeMax": 5002,
        "ThreadsBlocked": 25,
        "ThreadsRemaining": 4
    }
}
```

Table 3: Ranger RAZ JVM Metrics Decriptions

Metric	Description
GcTimeTotal	Total time taken by JVM Garbage Collector.
GcTimeMax	Maximum time taken by a single JVM Garbage Collection process.
GcCountTotal	Number of times JVM Garbage Collector runs.
SystemLoadAvg	Average load on the host on which the service is running.
ThreadsBusy	Total JVM threads which are busy.
ThreadsWaiting	Total JVM threads which are waiting.
ThreadsBlocked	Total JVM threads which are blocked.
ThreadsRemaining	Total remaining (not busy/ waiting/ blocked) JVM threads.
MemoryMax	Maximum memory (RAM) JVM can get from the host.
MemoryCurrent	Currently occupied memory (RAM) by JVM.
ProcessorsAvailable	Currently available processor cores on host.

Ranger REST API documentation

The Ranger Admin Web UI includes REST API documentation describes the framework used to enable, monitor and manage comprehensive data security across the Hadoop platform. The Ranger REST API is published as html, using the Swagger UI, at a url that has the following format:

http://<cdp-host-name>.root.hwx.site:6080/apidocs/swagger.json

You can access the Ranger API documentation using the Swagger user interface from Cloudera Manager. From CM, go to Ranger Ranger Admin Web UI .

To view the Ranger REST API documentation:

1. Log in to the Ranger Admin Web UI.
2. Open the Ranger Admin menu.
3. Select API Documentation.

Ranger Rest API documentation displays as html output in your web browser.