

Working with Admin APIs

Date published: 2020-10-30

Date modified: 2024-10-30



Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Enabling admin API support.....	4
Admin API demo page.....	4
Setting up a session.....	4
Retrieving or changing logging level.....	5

Enabling admin API support

By default, admin API URL support is disabled Cloudera Data Visualization. A platform administrator can enable this support, at the level of individual data types, through site-specific settings.

Procedure

1. Open Site Settings.
2. Scroll to Advanced Settings at the bottom of the left navigation.
3. To enable specific data types, add them to the assignment line.

See the example below:

Option

`ADMIN_API_URL_LIST = ['visuals', 'datasets']` Add the data types to enable visuals and datasets.

`ADMIN_API_URL_LIST = ['visuals', 'datasets', 'connections', 'users', 'groups', 'roles', 'segments', 'filterassociations']` To enable all data type support, list them all in the text box.

`ADMIN_API_URL_LIST = ['*']` Use the wildcard to specify all options.

4. Click SAVE.

Admin API demo page

This demo page serves as a practical tool for understanding how to interact with the Cloudera Data Visualization admin APIs. It provides a comprehensive view of how network requests are formed for accessing the admin APIs.

Procedure

1. Open Site Settings.
2. Scroll down to the Advanced Settings located at the bottom of the left navigation pane.
3. Add the following configuration setting.

```
ADMIN_API_DEMO_LIST = ['visuals', 'datasets', 'connections', 'users', 'groups', 'roles', 'segments', 'workspaces', 'filterassociations']
```

4. Click SAVE.

Results

Once this configuration is enabled, the Cloudera Data Visualization page is available on `[***domain***/arc/apps/apidemo]`.

Setting up a session

Cloudera Data Visualization Admin API supports two alternatives for setting up a session: standard login access and APIKey access.

Standard login access

To establish a session with username/password login, use the following code:

```
username = "" #(user name as obtained from external source)
password = "" #(password as obtained from external source)
session = requests.session()
response = session.get(login_url)
session.headers['referer'] = response.url
payload = {'username':username, 'password':password, 'csrfmiddlewaretoken':
session.cookies['arccsrftoken']}
session.post(login_url, data = payload)
```



Note: The login URL has the form [http|https]://[***host***]:[***port***]/arc/apps/login.

APIKey access

To establish a session through the Cloudera API Key system and avoid the login process, use the following code:

```
apikey = (apikey string)
session = requests.session()
session.headers['AUTHORIZATION'] = 'apikey %s' % apikey
```

In this approach, the client code must obtain the Cloudera Data Visualization access key through a client-controlled mechanism, then add the APIKey to the request header and avoid explicit login.

Fetching data from all datasets in the system

After setting up a session, you can fetch the entire data from all datasets in the system by using the following code:

```
response = session.get(api_url + 'datasets?detail=1')
datasets = response.json()
```



Note: The API URL has the form [http|https]://[***host***]:[***port***]/arc/adminapi/[***version***]. Use the URL option 'detail=true' to fetch all information in the GET call.

Retrieving or changing logging level

Cloudera Data Visualization Admin API provides a convenient way to manage logging levels for specific loggers, allowing you to control the verbosity of log messages as needed. It supports retrieving and changing the logging level for a specified logger within the system.

About this task

You can query the current logging level for a logger or modify it using the following API endpoints.

- GET: /arc/adminapi/loglevel/<logger-name>
- POST: /arc/adminapi/loglevel/<logger-name>

POST parameters

The POST endpoint expects a JSON document in the request body. The JSON document should contain a single field, level, specifying the desired log level. It accepts one of the predefined names in the Python logging module:

- CRITICAL
- DEBUG
- ERROR
- FATAL
- INFO

- WARN
- WARNING

Authentication

Both API key and session-based authentication are accepted. The retrieval of the logging level does not require any special permissions. However, updating the logging level requires the `sys_viewlogs` permission.

Procedure

1. Use the `curl` command to send a GET request to the specified endpoint to retrieve the current logging level for a specific logger.

```
curl -H 'content-type: application/json' -H 'authorization:
apikey [***API-KEY***]' [***VIZ_URL***]/arc/adminapi/loglev
el/[***LOGGER_NAME***]
```

Make sure to replace `[***API-KEY***]` with your actual API key, `[***VIZ_URL***]` with the Cloudera Data Visualization URL, and `[***LOGGER_NAME***]` with the name of the specific logger you want to check.

After executing the command, you receive a JSON response indicating the logger name and its current logging level.

2. Review the 'level' field in the response to determine the current logging level for the specified logger.
3. Use the `curl` command to send a POST request to the specified endpoint to change the logging level for a specific logger.

Provide the desired log level in the request body.

```
curl -H 'content-type: application/json' -H 'authorization:
apikey [***API-KEY***]' [***VIZ_URL***]/arc/adminapi/loglev
el/[***LOGGER_NAME***]
-d '{"level": "DEBUG"}'
```

After executing the command, you receive a JSON response confirming the changes, including the logger name and the new logging level.

4. Verify that the 'level' field in the response now reflects the updated logging level for the specified logger.

Example

Getting the current loglevel for the arcweb logger:

1. Request

```
curl -H 'content-type:application/json' -H 'authorization: apikey <API-K
EY>'
    <VIZ_URL>/arc/adminapi/loglevel/arcweb
```

2. Response

```
{
  "logger": "arcweb",
  "level": "INFO"
}
```

Setting the log level to DEBUG for arcweb:

1. Request

```
curl -H 'content-type:application/json' -H 'authorization: apikey <API-K
EY>' <VIZ_URL>/arc/adminapi/loglevel/arcweb -d '{"level": "DEBUG"}'
```

2. Response

```
{  
  "logger": "arcweb",  
  "level": "DEBUG"  
}
```