

..

Admin API

Date published: 2020-10-30

Date modified: 2022-09-21

CLOUDERA

<https://docs.cloudera.com/>

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

Admin API.....	4
Admin API Syntax and General Usage.....	5
Admin API Syntax Parameters.....	6
Python Admin API Data Format and Response.....	7
CURL Data Format and API Key Examples.....	7
Admin API Demo.....	8
Using Admin API Demo: Examples.....	9
Data type details.....	13
Users.....	14
Groups.....	14
Roles.....	15
Segments.....	17
Filter Associations API.....	18
Workspaces.....	19

Admin API

provides URL access to ArcViz server objects, such as users, groups, roles, datasets, segments, filter associations, connections, and visuals. This allows you to automate deployment operations, such as creating and managing roles, without logging into the ArcViz server console for access to command-line utilities.

Permissions

When running the Admin API interface, enforces the same security rules as when using the graphical user interface, ArcViz. For example, a user must have Manage roles and users system-level permissions to create new users or update existing users.

Admin API is CRUD

We designed the Admin API for simple object access: Create, Read, Update, and Delete. Hence, CRUD. It does not support high-level operations, such as adding tables to the dataset.

Admin API articles

- [Enabling Admin API Support](#)
- [Admin API syntax and general usage](#)
- [Admin API syntax parameters](#)
- [Setting Up a Session](#)
- [Python admin API data format and response examples](#)
- [CURL data format and API key examples](#)
- [Admin API demo](#)
 - [Examples](#)
- [Data type details](#)
 - [Users](#)
 - [Groups](#)
 - [Roles](#)
 - [Segments](#)
 - [Filter Associations API](#)
 - [Workspaces](#)

Related Information

[Admin API Syntax and General Usage](#)

[Admin API Syntax Parameters](#)

[Python Admin API Data Format and Response](#)

[CURL Data Format and API Key Examples](#)

[Admin API Demo](#)

[Using Admin API Demo: Examples](#)

[Data type details](#)

[Users](#)

[Groups](#)

[Roles](#)

[Segments](#)

[Filter Associations API](#)

[Workspaces](#)

Admin API Syntax and General Usage

The Admin API has a consistent pattern for each data type.

The Admin API has the following basic access syntax:

```
[http | https] : /host:port/arc/adminapi/version/data_type[/object_id][?options]
```

The parameters of this line are in Admin API syntax parameters.

HTTP Operations

The HTTP method specifies the following operation types:

GET

List of an item identified through the `object_id` or `object_name`, or all items, with default summary information. The URL option 'detail=true' returns all data details.

POST

Update: The fields in the request data update the item with matching `object_id`.

Create: If the URL or request data does not specify the `object_id`, ArcViz creates a new data item.

Validate: To verify that the connection is successful, issue the POST command two times:

1. Issue the POST command with the validate flag set to true.

```
[
  {
    "id":18,
    "name": "ArcEngine Dev",
    "type": "arcengine",
    "validate": "true",
    "info": {
      "PARAMS": {
        "HOST": "localhost",
        "PORT": "21051",
        "USERNAME": "admin",
        "SETTINGS": {
          "ANALYTICAL_VIEW_MAX_REFRESH_THREADS": "1",
          "MAX_PARTITIONS_FOR_REFRESH_INSERT": "1"
        }
      }
    }
  }
]
```

2. On success, issue the same POST command without the validate flag. This step saves the data.

DELETE

Delete the specified item.

HTTP Access Through Python

While you can use all standard HTTP access methods, we recommend the python request modules approach for HTTP operations. Note the following common setup:

```
import json
import requests
api_url = [http|https]://host:port/arc/adminapi/version
login_url = [http|https]://host:port/arc/apps/login
```

Related Information

[Admin API Syntax Parameters](#)

Admin API Syntax Parameters

The Admin API has a consistent pattern for each data type.

The Admin API has the following basic access syntax:

```
[http | https]://host:port/arc/adminapi/version/data_type[/object_id][?options]
```

host

The host of the ArcViz instance.

port

The port of the ArcViz instance.

version

The current API version is v1. This increments if the item data format changes. Whenever possible, we intend to support older versions for backward compatibility.

data_type

One of the ArcViz artifacts: users, groups, roles, datasets, connections, visuals, segments, filter associations, or workspaces.

object_id

The id of the individual object, such as a specific user, visual, or a specific dataset. You can either use the object_id or the object_name in the syntax, not both.

object_name

The name of the individual object, such as a specific user, visual, or a specific dataset. You can either use the object_id or the object_name in the syntax, not both.

options

Further request options, such as level of information details when 'detail=true'.

api_url

The address of the API management system, in the form [http|https]://host:port/arc/adminapi/version.

login_url

To authenticate the end user, address of the login URL, in the form [http|https]://host:port/arc/apps/login.

Python Admin API Data Format and Response

provides examples of python Admin API data format and response.

The response data for GET operations is a list of JSON items. For POST operations, such as UPDATE and CREATE, the input format is a structure with a data field that contains the JSON list of items. The UPDATE and CREATE operations process one item at a time, so the list is exactly one entry long.

The response data for POST operations is the updated item, matching a GET with detail=1 for the item, as demonstrated in [Example 1: Setting the name for role ID=1 to 'System Admin'](#) on page 7.

For item updates, it is only necessary to specify the fields you are updating. merges the supplied fields in the input data with the existing item's data, as demonstrated in [Example 2: Checking role ID=2; updating by adding a new user](#) on page 7.

Example 1: Setting the name for role ID=1 to 'System Admin'

```
payload = {'name': 'System Admin'}
session.post(api_url + '/roles/1', data={'data': json.dumps([payload])})
```

Note that the API URL has the following form:

```
[http|https]://host:port/arc/adminapi/version
```

For syntax of other parameters, see Admin API syntax parameters.

Example 2: Checking role ID=2; updating by adding a new user

```
response = session.get(api_url + '/roles/2?detail=1')
role = response.json()[0]
if 'new_user' not in role['users']:
    payload = {'users': role['users'] + ['new_user']}
    session.post(api_url + '/roles/2', data={'data': json.dumps([payload])})
```

For the definition of fields for each data type, see Data type details.

Related Information

[Admin API Syntax Parameters](#)

[Data type details](#)

CURL Data Format and API Key Examples

provides examples of API Key in CURL data format.

When you add the APIKey to the request header and avoid explicit login, all interactions become simpler. The examples in this article use an APIKey obtained through the Manage API Keys interface, on the host:port/arc/apps/apikeys browser page of the DataViz installation. The actual APIKey and the method of retrieving the key depends on the user system.

See [Example 1](#) to learn how to get all roles information, and [Example 2](#) to change the description of a role.

Example 1: Getting all roles

To use CURL to obtain a full dump of all roles, use the following command.

Note that the output is piped to the standard python JSON dumper for easier reading; it is not necessary for CURL access.

```
curl -s \
-X GET \
-H "Authorization: apikey ApiKey" \
api_url/roles?detail=1> | python -m json.tool
```

Note that the login URL has the form [http|https]://host:port/arc/apps/login. For syntax of other parameters, see Admin API syntax parameters.

Example 2: Changing the role description

To change the description for role ID 3, use the following command.

For CURL to supply data through the POST method, use the application/x-www-form-urlencoded content type.

```
curl -s \
-X POST \
-H "Content-Type: application/x-www-form-urlencoded" \
-H "Authorization: apikey ApiKey" \
-d 'data=[{"desc": "Updated description again"}]' \
api_url/roles/3
```

For the definition of fields for each data type, see Data type details. For syntax of other parameters, see Admin API syntax parameters.

Related Information

[Admin API Syntax Parameters](#)

[Data type details](#)

Admin API Demo

We are including a simple GUI demo to demonstrate how the URL-based Admin API can easily display the data format for each item type.



Note:

- Use the demo with some caution: the system does not confirm DELETE operations, and it is not meant for implementing system configuration changes. It is simply a working example of the Admin API.
- The demo is not enabled by default. It must be enabled, much as the actual Admin API URL support must be enabled, for each item.

Use the following statement to enable the demo:

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

Alternatively, use the wild card to specify all options:

```
ADMIN_API_DEMO_LIST = ['*']
```

To fully enable all APIs and all demo tabs, use the wild card character in the settings entries for both ADMIN_API_DEMO_LIST and ADMIN_API_URL_LIST:

```
ADMIN_API_DEMO_LIST = ['*']
ADMIN_API_URL_LIST = ['*']
```


After enabling the demo, you can access it through the following URL:

```
host_path/arc/apps/apidemo
```

Using Admin API Demo: Examples

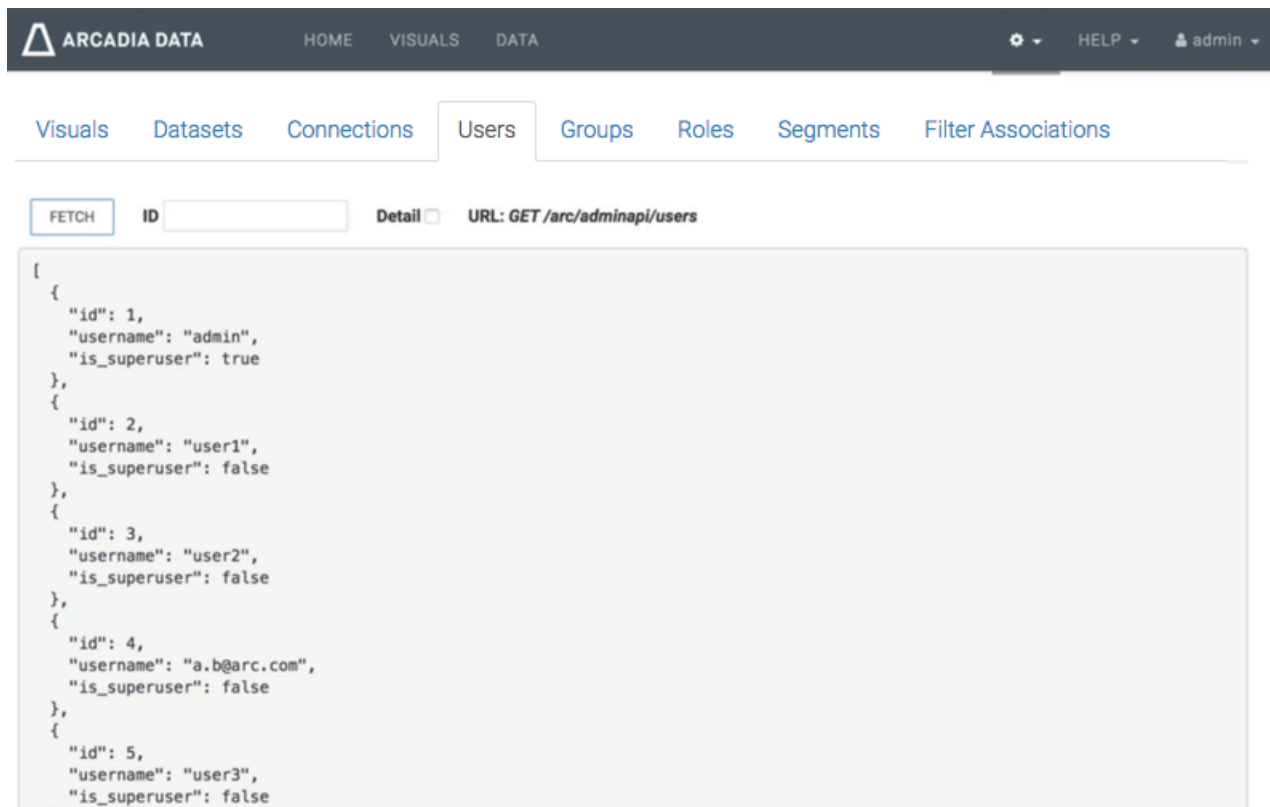
's URL-based Admin API can easily display the data format for each item type.

The demo has a tab for each enabled item type, and supports all the API functions: Fetch (one), Fetch All, Create, Update, and Delete.

The following examples demonstrate some of the functions available through the demo.

Fetching All User Information

To get information on all users, simply click Fetch. returns the list of registered users, as demonstrated by the following screenshot.



Fetching Single User Information, With Detail

The following steps shows how to extract information on a single item using its ID or name. The following image is a screen shot of extracting information on a single item using its ID, and contains the actions available for single items: Update, Clone, and Delete.

- Under the Users tab, enter the ID.
- Select the Detail option to get the full data for that item, in this case user ID 1.
- Click Fetch.

The screenshot shows the Arcadia Data Admin API interface. The top navigation bar includes the Arcadia Data logo, 'HOME', 'VISUALS', and 'DATA'. A user profile 'admin' is visible in the top right. The main navigation menu includes 'Visuals', 'Datasets', 'Connections', 'Users', 'Groups', 'Roles', 'Segments', 'Filter Associations', and 'Workspaces'. The 'Users' tab is active. Below the navigation, there are buttons for 'FETCH', 'UPDATE', 'CLONE', and 'DELETE'. A search input field contains 'ID/Name 1' and is highlighted with a green border. To the right of the search field is a 'Detail' checkbox that is checked. Below the search field, the URL is displayed as 'URL: GET /arc/adminapi/v1/users/1?detail=1'. The main content area shows a JSON response for the user with ID 1.

```
[
  {
    "id": 1,
    "username": "admin",
    "is_superuser": true,
    "is_active": true,
    "date_joined": "2018-02-17 03:39:34 UTC",
    "last_login": "2018-09-04 22:55:14 UTC",
    "first_name": "",
    "last_name": "",
    "groups": [
      {
        "id": "",
        "name": ""
      }
    ],
    "roles": []
  }
]
```

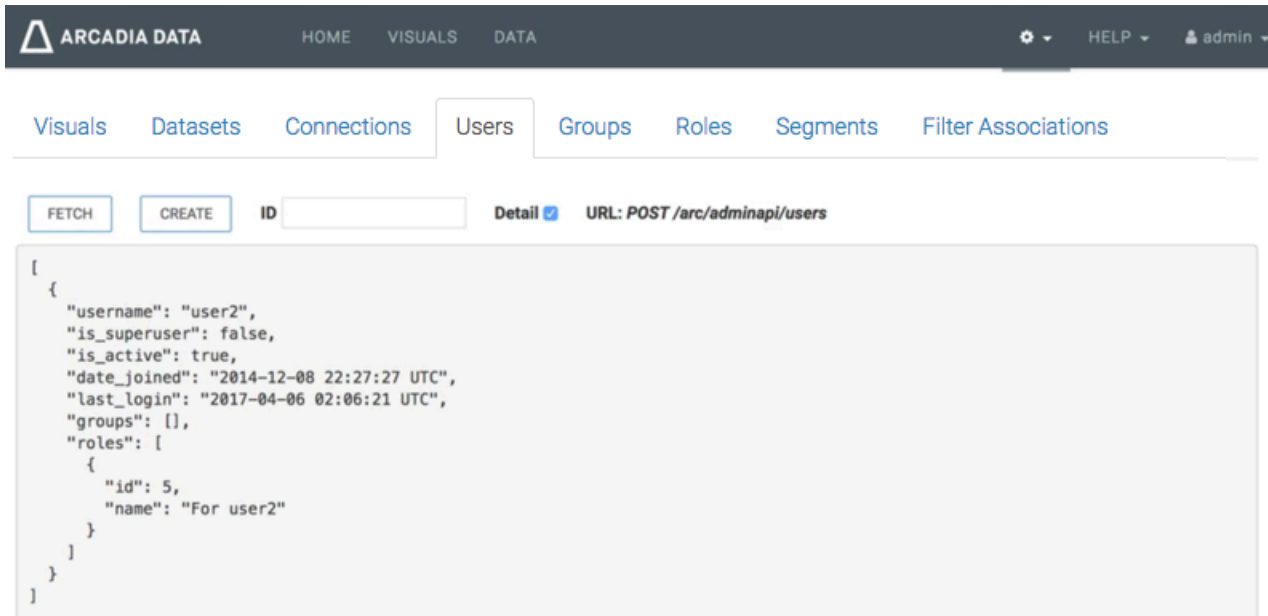
The following image is a screen shot of extracting information on a single item using its name. In this case name is admin.

The screenshot shows the Arcadia Data Admin API interface. The top navigation bar includes the Arcadia Data logo, 'HOME', 'VISUALS', and 'DATA'. A user profile 'admin' is visible in the top right. The main navigation menu includes 'Visuals', 'Datasets', 'Connections', 'Users', 'Groups', 'Roles', 'Segments', 'Filter Associations', and 'Workspaces'. The 'Users' tab is active. Below the navigation, there are buttons for 'FETCH', 'UPDATE', 'CLONE', and 'DELETE'. A search input field contains 'ID/Name admin' and is highlighted with a green border. To the right of the search field is a 'Detail' checkbox that is checked. Below the search field, the URL is displayed as 'URL: GET /arc/adminapi/v1/users?name=admin&detail=1'. The main content area shows a JSON response for the user with name 'admin'.

```
[
  {
    "id": 1,
    "username": "admin",
    "is_superuser": true,
    "is_active": true,
    "date_joined": "2018-02-17 03:39:34 UTC",
    "last_login": "2018-09-04 22:55:14 UTC",
    "first_name": "",
    "last_name": "",
    "groups": [
      {
        "id": "",
        "name": ""
      }
    ],
    "roles": []
  }
]
```

Cloning an Item

When you Clone an item, the resulting screen shows a duplicate of the item, but clears the ID field. This is a partial operation; see [Creating a New Item](#) on page 11.



ARCADIA DATA HOME VISUALS DATA HELP admin

Visuals Datasets Connections **Users** Groups Roles Segments Filter Associations

FETCH CREATE ID Detail URL: POST /arc/adminapi/users

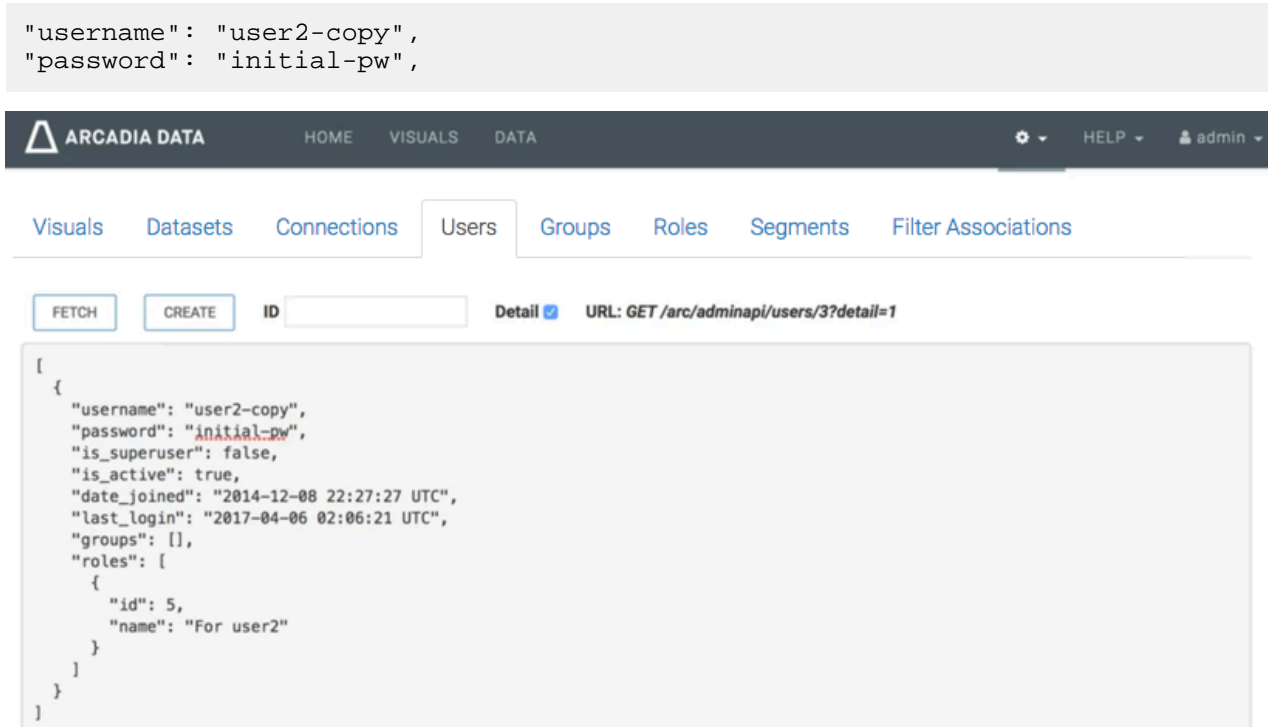
```
[
  {
    "username": "user2",
    "is_superuser": false,
    "is_active": true,
    "date_joined": "2014-12-08 22:27:27 UTC",
    "last_login": "2017-04-06 02:06:21 UTC",
    "groups": [],
    "roles": [
      {
        "id": 5,
        "name": "For user2"
      }
    ]
  }
]
```

Creating a New Item

If you click Create at this time (after [Cloning an Item](#) on page 11), returns an error, because a user with this name already exists.

When you change the username, you must also supply the initial password. Here, we created a new user by changing the username and adding an extra line that specifies the password:

```
"username": "user2-copy",
"password": "initial-pw",
```



ARCADIA DATA HOME VISUALS DATA HELP admin

Visuals Datasets Connections **Users** Groups Roles Segments Filter Associations

FETCH CREATE ID Detail URL: GET /arc/adminapi/users/3?detail=1

```
[
  {
    "username": "user2-copy",
    "password": "initial-pw",
    "is_superuser": false,
    "is_active": true,
    "date_joined": "2014-12-08 22:27:27 UTC",
    "last_login": "2017-04-06 02:06:21 UTC",
    "groups": [],
    "roles": [
      {
        "id": 5,
        "name": "For user2"
      }
    ]
  }
]
```

When you click Create now, notifies you that your update is successful, and refreshes the interface to show the results consistent with fetching detailed information for a specific user, as described in [Fetching Single User Information, With Detail](#) on page 9.

Note that for security reasons, the password is not included in the fetched user details.

The screenshot shows the Arcadia Data Admin API interface. At the top, there is a navigation bar with 'ARCADIA DATA', 'HOME', 'VISUALS', 'DATA', 'HELP', and a user profile 'admin'. Below the navigation bar, there are tabs for 'Visuals', 'Datasets', 'Connections', and 'Filter Associations'. A green success notification box is displayed, stating 'Success Update Success!'. Below the notification, there are buttons for 'FETCH', 'UPDATE', 'CLONE', and 'DELETE'. The 'UPDATE' button is highlighted. To the right of the buttons, there is a text input field containing 'ID 12' and a 'Detail' checkbox which is checked. Below the buttons and input field, there is a text area displaying a JSON response for user ID 12:

```
[
  {
    "id": 12,
    "username": "user2-copy",
    "is_superuser": false,
    "is_active": true,
    "date_joined": "2017-04-09 00:39:59 UTC",
    "last_login": "2017-04-09 00:39:59 UTC",
    "groups": [],
    "roles": [
      {
        "id": 5,
        "name": "For user2"
      }
    ]
  }
]
```

Changing Passwords

To change the user's password, you must supply both the current password (password) and the new password (new_password).

Edit the user detail by adding the following two lines of code, and click Update.

```
"password": "initial-pw",
"new_password": "updated-pw",
```

```

[
  {
    "id": 12,
    "username": "user2-copy",
    "password": "initial-pw",
    "new_password": "updated-pw",
    "is_superuser": false,
    "is_active": true,
    "date_joined": "2017-04-09 00:39:59 UTC",
    "last_login": "2017-04-09 00:39:59 UTC",
    "groups": [
      {
        "id": 6,
        "name": "Clone of group Test-D"
      }
    ],
    "roles": [
      {
        "id": 5,
        "name": "For user2"
      }
    ]
  }
]

```

Data type details

provides URL access to ArcViz server objects.

The Admin API uses specific JSON definitions for each Data Type:

- Users
- Groups
- Roles
- Segments
- Filter associations API
- Workspaces

Note that we deliberately chose not to document the details of creating datasets, connections, and visuals. They are all highly complex structures, and should be created directly in the application, through the graphical user interface.

For the GET requests, many returned fields are only informational, and cannot be updated through subsequent POST requests.

Some fields are themselves complex structures that contain sub-fields. The update logic that merges the supplied input fields with the existing data applies to the top-level fields only. For example, the role type contains a privs field, which is a list of individual privilege records. To update the privs field, you must supply the entire list, not merely the individual list element.

Related Information

[Users](#)

[Groups](#)

[Roles](#)

[Segments](#)

[Filter Associations API](#)

Workspaces

Users

provides URL access to the ArcViz server object, users.

When creating a new user, you must supply the password field. You must also supply the password field when updating a user's password. Like in the GUI, a regular user (non-admin) can only change their own password, and must supply the current password in the `old_password` field.

Supplying None (or null in the API demo page, the javascript version of None) for the password makes the user account unusable for login.

For the list of groups and roles, the name is supplied for information only. When updating the users's groups or roles, only the ID fields are necessary.

The examples in this article use an API Key obtained through the Manage API Keys interface, on the `host:port/arc/apps/apikeys` browser page of the DataViz installation. The actual API Key and the method of retrieving the key depends on the user system.

Here is a CURL example for setting the roles for user ID=3 to IDs 5, 7, and 8.

```
curl -s \
  -X POST \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -H "Authorization: apikey api_key" \
  -d 'data=[{"roles": [{"id":5}, {"id":7}, {"id":8}]}]' \
  api_url + '/users/3'
```

Note that the API URL has the form `[http|https]://host:port/arc/adminapi/version`.

The JSON fields for user's data type are defined as follows:

Table 1: JSON Fields for Users Data Type

Field	Detail Only	Updatable	Description
id	No	No	User ID
username	No	Yes	Username, limited to alphanumeric, period, underscore, and dash
is_superuser	No	No	Indicates the admin userm, who has full permissions
is_active	Yes	No	TRUE: user can long through the UI FALSE: user cannot login through the UI, but trusted authentication works
date_joined	Yes	No	Shows the creation date for this user's metadata entry
last_login	Yes	No	Shows the last login date for this user
groups	Yes	Yes	List of groups to which this user belongs; each entry shows group ID and group name
roles	Yes	Yes	List of roles to which this user belongs; each entry shows role ID and role name

Groups

provides URL access to the ArcViz server object, groups.

Like the users data type, the names for list of groups and roles are supplied for information only. When updating the users's groups or roles, only the ID fields are necessary.

The JSON fields for group's data type are defined as follows:

Table 2: JSON Fields for Groups Data Type

Field	Detail Only	Updatable	Description
id	No	No	Group ID
name	No	Yes	Group name
users	Yes	Yes	List of users in this group; each entry shows the user id and username
roles	Yes	Yes	List of roles to which this group belongs; each entry shows role ID and role name

Roles

provides URL access to the ArcViz server object, roles.

To support installations that store the users and groups information outside the ArcViz environment (such as LDAP), the role membership lists for users and groups only store names. During role update and create operations, ArcViz accepts the user and group names 'as is', without validating them.

Each entry in the privs list corresponds to a single privilege row in the ArcViz role edit screen. Each row contains fields for the privilege type (ptype), an identifier section, and a list of permissions (perms) for the identified objects, such as datasets or data connections. Each privilege type has a specific identifier, and set of possible permissions. ArcViz stores the dataset IDs and connection IDs within the identifier sections as a STRING, and uses the special value "-1" to indicate "All datasets" or "All connections".

This article includes the following topics:

- [Defining Roles Data Type](#) on page 15
- [Privileges Types](#) on page 15
- [Creating Roles](#) on page 16

Defining Roles Data Type

The JSON fields for role's data type are defined as follows:

Table 3: JSON Fields for Roles Data Type

Field	Detail Only	Updatable	Description
id	No	No	Role ID
name	No	Yes	Role name
desc	No	Yes	Role description
users	No	Yes	List of usernames that belong to this role
groups	No	Yes	List of groups that belong to this role
privs	Yes	Yes	List of privilege structures for this role, as described in Privileges Types on page 15

Privileges Types

The Role-Based Access Control system supports the following permission types:

- *ptype: "system"*

Identifier

None

Permissions

Permission Name	Description
sys_editperm	Manage roles and users
sys_styles	Manage styles and settings
sys_viewlogs	View query logs
sys_editconn	Manage data connections

- *ptype: "dataconn"*

Identifier

Field Name	Description	Example
dclist	List of data connection IDs, or -1 for 'All data connections'	"dclist" : ["-1"]

Permissions

Permission Name	Description
dc_aviews	Manage analytical views
dc_upload	Import data
dc_expore	Create datasets and explore tables

- *ptype: "dataset"*

Identifier

Field Name	Description	Example
dcid	Data connection ID for this privilege, or -1 for 'All'	"dcid" : "-1"
dslist	List of dataset IDs for this privilege	"dslist" : ["1", "2", "3"]

Permissions

Permission Name	Description
dc_aviews	Manage analytical views
dc_upload	Import data
dc_expore	Create datasets and explore tables

Creating Roles

The following code creates a new role with groups `dataconn_managers` and `arcviz_admins`. The role has system-level permissions to view logs, and to create new datasets. It also has full permissions on all connections and all datasets.

The actual API Key and the method of retrieving the key depends on the user system.

```
curl -s \
-X POST \
-H "Content-Type: application/x-www-form-urlencoded" \
-H "Authorization: apikey api_key" \
-d 'data=[{
  "name": "Connection manager",
  "desc": "Data connection management",
  "groups": ["dataconn_managers", "arcviz_admins"],
```



```

    "privs": [
      { "ptype": "system",
        "perms": ["sys_viewlogs", "sys_editconn"]
      },
      { "ptype": "dataconn",
        "dclist": ["-1"],
        "perms": ["dc_aviews", "dc_upload", "dc_explore"]
      },
      { "ptype": "dataset",
        "dcid": "-1",
        "dslist": ["-1"],
        "perms": ["ds_manage", "ds_appedit", "ds_appview"]
      }
    ]
  }
}]]' \
127.0.0.1:7999/arc/adminapi/roles

```

When viewing this role through the ArcViz user interface, it appears on the edit screen like this:

Role: Connection manager

SAVE UNDO

Name: Connection manager

Description: Data connection management

Privileges Members

Component	Type		Create workspaces	Manage roles and users	Manage site settings	Manage custom styles	Manage jobs, email templates	View activity logs	Manage data connections	Grant manage dataset	Grant manage dashboards	Grant view dashboards	Manage analytical views	Import data	Create datasets, explore tables	Manage dataset	Manage dashboards	View dashboards
System		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>									
All connections		<input checked="" type="checkbox"/>									<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					<input type="checkbox"/>
All connections / All datasets		<input checked="" type="checkbox"/>													<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

+ ADD PRIVILEGE

Segments

provides URL access to the ArcViz server object, segment.

The segment data field is a complex structure that matches the UI Segment edit screen.

This article includes the following topics:

- [Segment Data Type](#) on page 17
- [Data Field Detail](#) on page 18

Segment Data Type

The JSON fields for role's data type are defined as follows:

Table 4: JSON Fields for Segment Data Type

Field	Detail Only	Updatable	Description
id	No	No	Segment ID
name	No	Yes	Segment name
dataset_id	No	No	Dataset ID for the segment
created	Yes	No	Shows the creation date for this segment
created_by	Yes	No	Username of the segment creator
updated	Yes	No	Shows the most recent update for this segment
updated_by	Yes	No	Username of the segment updater
data	Yes	Yes	Segment definition data, as described in Data Field Detail on page 18

Data Field Detail

The segment data field is a complex structure with the following specification:

Field	Description
entities	List of dimension expressions emitted by the entity-type segments
group	Name of the group to which this segment belongs
filters	List of filter expressions that define this segment
applyToNewVisuals	Specify if new visuals on the dataset should start with filters defined in this segment

Filter Associations API

provides URL access to the ArcViz server object, filter association.

The filter association data field is a complex structure that matches the UI Filter Association edit screen.

This article includes the following topics:

- [Segment Data Types](#) on page 18
- [Data Field Detail](#) on page 19

Segment Data Types

The JSON fields for role's data type are defined as follows:

Table 5: JSON Fields for Filter Association Data Type

Field	Detail Only	Updatable	Description
id	No	No	Filter association ID
name	No	Yes	Filter association name
dataset_id	No	No	Dataset ID for the filter association
created	Yes	No	Shows the creation date for this filter association
created_by	Yes	No	Username of the filter association creator
updated	Yes	No	Shows the most recent update for this filter association
updated_by	Yes	No	Username of the filter association updater

Field	Detail Only	Updatable	Description
users	Yes	Yes	List of user IDs to which the filter association applies
groups	Yes	Yes	List of group IDs to which this filter association applies
data	Yes	Yes	List of segments that make up this filter association, as described in Data Field Detail on page 19

Data Field Detail

The filter association data field is a complex structure with the following specification:

Field	Description
id	ID of segment applied to filter association
group	Name of the group to which the identified segment belongs
negate	Indicates that the filter association defines the rows NOT IN the segment, rather than IN the segment

Workspaces

provides URL access to the ArcViz server object, workspaces.

In addition to the standard public workspace that all users share, and the single private workspace that each user has, users with Create Workspace privilege can create custom workspaces. These workspaces may be shared by specific users and user groups, so they can develop and view dashboards that are inherently useful to their line of business. Within each of these workspaces, each user has a defined access level: View Only, Edit, or Manage.

This article includes the following topics:

- [Workspace Data Types](#) on page 19
- [Access Control List in Workspaces](#) on page 20
- [Creating Workspaces](#) on page 20

Workspace Data Types

The JSON fields for workspace data type are defined as follows:

Table 6: JSON Fields for Workspace Data Type

Field	Detail Only	Updatable	Description
id	No	No	Workspace ID
name	No	Yes	Workspace name
desc	No	Yes	Workspace description
editable	Yes	No	Permission to update only a non system-managed workspace (Custom workspace). The system managed workspaces are Public and Private workspaces.
private_user_id	Yes	No	ID of the user for private workspaces
acl	Yes	Yes	Access Control List (ACL) for a workspace

Access Control List in Workspaces

The workspace ACL is a list of privilege entries. Each entry contains three items. The following acl syntax shows two entries:

```
"acl": [
  [entry_type, access_level, group_name],
  [entry_type, access_level, user_name]]
```

Entry	Encoding
entry_type	1 = User, 2 = Group
access_level	1 = View, 2 = Edit, 3 = Manage
user_name/group_name	User or group name of the entry_type

Creating Workspaces

The following code creates a new workspace Test workspace and provides View access to a special group Everyone and Manage access to user admin.

The actual APIKey and the method of retrieving the key depends on the user system.

```
curl -s \
-X POST \
-H "Content-Type: application/x-www-form-urlencoded" \
-H "Authorization: apikey apikey" \
-d 'data=[{
  "name": "Test workspace",
  "desc": "Workspace created via admin api",
  "acl": [[2, 1, "Everyone"], [1, 3, "admin"]]
}]' \
127.0.0.1:7999/arc/adminapi/workspaces
```

When viewing this workspace through the ArcViz GUI, it appears on the workspace edit modal window like this:

Create Workspace







Name

Test Workspace

Description

Workspace created via admin api

 type username	 type group name, type 'Everyone' for all	ADD
---	--	-----

 admin	<input type="radio"/> View Only	<input type="radio"/> Edit	<input checked="" type="radio"/> Manage	
 Everyone	<input checked="" type="radio"/> View Only	<input type="radio"/> Edit	<input type="radio"/> Manage	

NOTE: Users must also have appropriate privileges to the underlying dataset(s)

 DELETE WORKSPACE	CLOSE	SAVE
---	-------	------