ReQUIAM

Release v1.0.1

Yan Han, UA Research Data Repository (ReDATA) Team

Jun 17, 2022

CONTENTS:

1 Overview **3**

2 Getting Started **5**
- 2.1 Requirements . . . 5
- 2.2 Installation Instructions . . . 5
- 2.3 Configuration Settings . . . 6
- 2.4 Testing Installation . . . 6

3 Execution **7**
- 3.1 Manual Changes . . . 7
- 3.2 API Management of Grouper Groups . . . 8

4 Versioning **9**

5 Continuous Integration **11**

6 Authors **13**

7 Changelog **15**

8 License **17**

9 API Documentation **19**
- 9.1 `requiam` package . . . 19

10 Indices and tables **31**

Python Module Index **33**

Index **35**

ReQUIAM | ReDATA EDS Query and Update for Identity and Access Management

Identity and access management software for ReDATA

CHAPTER

ONE

OVERVIEW

This identity and access management (IAM) software performs the following for the University of Arizona's Research Data Repository (ReDATA):

1. It conducts EDS queries to retrieve classification information (e.g., student, staff, and/or faculty) and association with organization codes (i.e., departments/colleges)
2. Based on classification information and primary organization association, it sets `ismemberof` Grouper membership

The Grouper memberships are as follow:

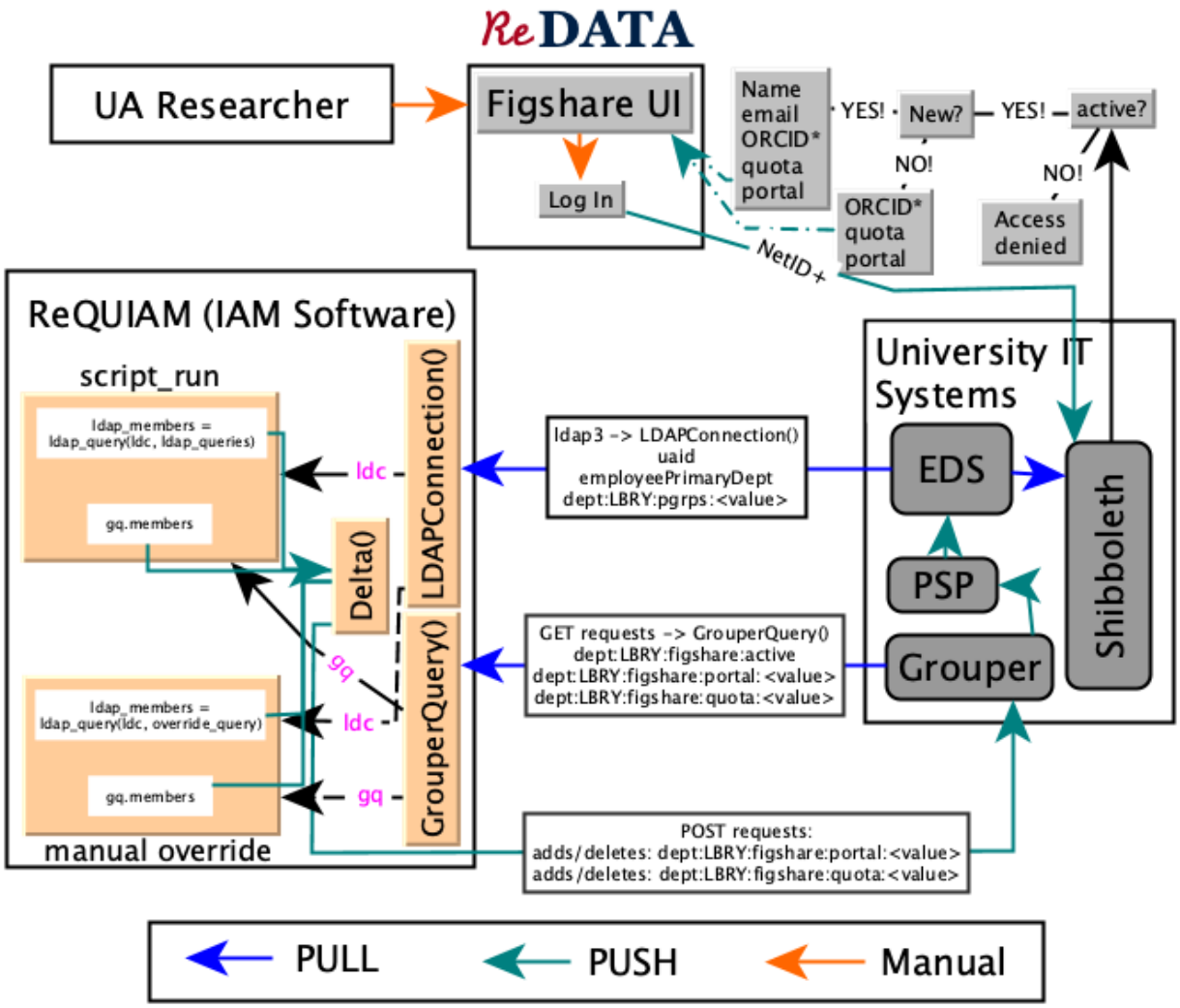
1. The allowed user quota for upload (in bytes), determined by the user's classification, is set by Grouper `figshare:quota:<value>` group
2. The "research theme/portal", determined by the user's organizational affiliation, is set by Grouper `figshare:portal:<value>` group

For the latter, these portals and their association with University organization code(s) are defined within this CSV file.

Note that access is granted to the service through membership in a Grouper `figshare:active` group. These memberships are done indirectly based on other Grouper membership set forth by University Library privileges.

This software is based on the existing patron software developed for the University of Arizona Libraries.

An illustration of the service architecture workflow is provided below.


ReDATA
UA Researcher
Figshare UI
Log In
Name
email
ORCID*
quota
portal
YES!
New?
YES!
active?
NO!
NO!
ORCID*
quota
portal
Access
denied
NetID+
ReQUIAM (IAM Software)
script_run
ldap_members =
ldap_query(ldc, ldap_queries)
gq.members
ldc
Delta()
LDAPConnection()
gq
ldap_members =
ldap_query(ldc, override_query)
ldc
gq.members
gq
GrouperQuery()
manual override
ldap3 -> LDAPConnection()
uaid
employeePrimaryDept
dept:LBRY:pgrps:<value>
GET requests -> GrouperQuery()
dept:LBRY:figshare:active
dept:LBRY:figshare:portal:<value>
dept:LBRY:figshare:quota:<value>
POST requests:
adds/deletes: dept:LBRY:figshare:portal:<value>
adds/deletes: dept:LBRY:figshare:quota:<value>
University IT
Systems
EDS
PSP
Grouper
Shibboleth
PULL
PUSH
Manual

CHAPTER

TWO

GETTING STARTED

These instructions will have the code running on your local or virtual machine.

2.1 Requirements

The requirements.txt indicates the required python libraries. In short, you will need the following to have a working copy of this software.

1. Python (>=3.7)
2. ldap3 (2.6.1)
3. numpy (1.20.0)
4. redata (>=0.3.2)
5. pandas (1.2.3)
6. tabulate (0.8.3)
7. requests (2.25.1)

2.2 Installation Instructions

2.2.1 Python and setting up a `conda` environment

First, install a working version of Python (>=3.7). We recommend using the Anaconda package installer.

After you have Anaconda installed, you will want to create a separate `conda` environment and activate it:

```
$ (sudo) conda create -n figshare_patrons python=3.7
$ conda activate figshare_patrons
```

Next, clone this repository into a parent folder:

```
(figshare_patrons) $ cd /path/to/parent/folder
(figshare_patrons) $ git clone https://github.com/UAL-RE/ReQUIAM.git
```

With the activated `conda` environment, you can install with the `setup.py` script:

```
(figshare_patrons) $ cd /path/to/parent/folder/ReQUIAM
(figshare_patrons) $ (sudo) python setup.py develop
```

This will automatically installed the required `pandas`, `ldap3`, `requests`, and `numpy` packages.

You can confirm installation via `conda list`

```
(figshare_patrons) $ conda list requiam
```

You should see that the version is `1.0.1`.

2.3 Configuration Settings

Configuration settings are specified through the config/figshare.ini file. The most important settings to set are those populated with `***override***`. However, for our scripts, these settings can be specified using multi-character flag options, such as `--ldap_password`. Note that most `figshare.ini` settings can be overwritten through the command line.

For manual override (v0.11.0) where IAM portal and quota settings differ from norm, `config` will include two CSV templates for portal and quota to specify those changes.

2.4 Testing Installation

To test the installation without performing any `portal` or `quota` query, execute the following command:

```
(figshare_patrons) $ export password="insert_password"
(figshare_patrons) $ export persist_path="/path/to/persistent/storage"
(figshare_patrons) $ ./scripts/script_run --config config/figshare.ini \
                       --persistent_path $persist_path \
                       --ldap_password $password --grouper_password $password
```

Test command-line flags (`test` and `test_reverse`) are available to test EDS query and Grouper synchronization (with the `sync` flag) by executing the following :

```
(figshare_patrons) $ ./scripts/script_run --test \
                       --config config/figshare.ini --persistent_path $persist_path \
                       --ldap_password $password --grouper_password $password --sync
```

Note that the above will add a test NetID account to the following Grouper group: `arizona.edu:dept:LBRY:figshare:test`

Without the `sync` flag, the above command line will perform a "dry run". It will indicate what Grouper updates will occur.

To undo this change, use the `test_reverse` flag:

```
(figshare_patrons) $ ./scripts/script_run --test_reverse \
                       --config config/figshare.ini --persistent_path $persist_path \
                       --ldap_password $password --grouper_password $password --sync
```

CHAPTER

THREE

EXECUTION

To execute the script and update Grouper and EDS, include the `portal`, `quota`, and `sync` command-line flags:

```
(figshare_patrons) $ ./scripts/script_run --config config/figshare.ini \
                       --persistent_path $persist_path \
                       --ldap_password $password --grouper_password $password \
                       --quota --portal --sync
```

Note: Without the `sync` flag, the above command line will perform a "dry run" where both `quota` and `portal` queries are conducted. It will indicate what Grouper updates will occur.

By default, changes occur on the `figshare` stem. Execution can occur on the `figtest` stem with the `--grouper_figtest` boolean flag.

There are additional options to run a subset of portals or organization codes. This is specified with the `--org_codes` or `--groups` options, which accepts comma-separated inputs. For this to work, the `--portal` must be set. If `--quota` is specified, those users are added to the appropriate group. Note that with this option, it will create and populate a `figtest:group_active` group that allows for indirect membership association. There are a couple of interactive prompts to create the `figtest:group_active` group or provide an existing one to use/update.

3.1 Manual Changes

While the primary use of this software is automated updates through Grouper, there are additional scripts for handling. One of those pertains to overriding default changes (e.g., a user's quota, involvement with a specific portal). To this end, the `user_update` script should be used. It has several features:

1. It can add a number of users to a specific group and also remove them from its previous group assignment(s)
2. It will update the appropriate CSV files. This ensures that the changes stay when the automated script `script_run` is executed
3. It has the ability to move a user to the "main" or "root" portal
4. It has a number of built-in error handling to identify possible input error. This includes:
    - A username that is not valid
    - A Grouper group that does not exist
    - Prevent any action if the user belongs to the specified group

Execution can be done as follows:

```
(figshare_patrons) $ ./scripts/user_update --config config/figshare.ini \
                       --persistent_path $persist_path \
                       --ldap_password $password --grouper_password $password \
                       --quota 123456 --portal testportal --netid <username> --sync
```

Here, the script will update the specified <username> to be associated with the 123456 quota and the testportal portal. Much like script_run, execution requires the --sync flag. Otherwise, a list of changes will be provided. Note: <username> can be a list of comma-separated users (e.g., user1,user2,user3) or a .txt file with each username on a new line.

```
user1
user2
user3
```

To remove a user from its current assignment and place it on the main portal, use: --portal root. For quota, the root option will remove any quota association (this is equivalent to a zero quota)

The manual CSV files are specified in the config file:

```
# Manual override files
portal_file = config/portal_manual.csv
quota_file = config/quota_manual.csv
```

These settings, much like other settings (see python requiam/user_update --help), can be overwritten on the command line:

```
--portal_file /path/to/portal_manual.csv
--quota_file /path/to/quota_manual.csv
```

Note that working templates are provided in the config folder for quota and portal.

To disable updating the the manual CSV files, you can include the following flags: --portal_file_noupdate --quota_file_noupdate

By default, changes occur on the figshare stem. Execution can occur on the figtest stem with the --grouper_figtest boolean flag.

3.2 API Management of Grouper Groups

The add_grouper_groups currently create and assign privileges to groups through the Grouper API. It uses the ReQUIAM_csv's CSV file for research themes and sub-portals. In addition, another Quota Google Sheet exists for the quotas. The script will check whether a group exists. If the add flag is provided, it will create the group and assign privileges for GrouperSuperAdmins and GrouperAdmins. If a group already exists, it will skip to the privilege assignments. To execute the script:

```
(figshare_patrons) $ ./scripts/add_grouper_groups --config config/figshare.ini \
                       --persistent_path $persist_path --grouper_password $password \
                       --main_themes --sub_portals --quota --add
```

The main_themes, sub_portals and quota flags will conduct checks and create those sets of groups. Without the add flag, it is a dry run. By default this works on a testing Grouper stem figtest. Set the production flag to implement on the production stem, figshare.

CHAPTER

FOUR

VERSIONING

We use SemVer for versioning. For the versions available, see the tags on this repository.

CHAPTER

FIVE

CONTINUOUS INTEGRATION

Initially we started using Travis CI; however, due to the change in pricing for open-source repositories, we decided to use GitHub Actions. Currently, there are two GitHub Action workflows:

1. A "Create release" workflow, create-release.yml, for new releases when a tag is pushed
2. A "Python package" workflow, python-package.yml, for builds and tests

CHAPTER

SIX

AUTHORS

- Yan Han, (@yhan818)
- Chun Ly, Ph.D. (@astrochun)
- Damian Romero (@damian-romero)

See also the list of contributors who participated in this project.

CHAPTER

SEVEN

CHANGELOG

See the CHANGELOG for all changes since project inception.

CHAPTER

EIGHT

LICENSE

This project is licensed under the MIT License - see the LICENSE file for details.

CHAPTER

NINE

API DOCUMENTATION

9.1 requiam package

9.1.1 Modules

commons module

requiam.commons.**dict_load**(*config_file*, *vargs=None*)

Read in a config INI file using `configparser` and return a `dict` with sections and options

Parameters

- **config_file** (`str`) – Full/relative path of configuration file
- **vargs** (`Optional[Dict[str, str]]`) – Command-line arguments from script

Return type `dict`

Returns Python `dict` of configuration settings

requiam.commons.**figshare_group**(*group*, *root_stem*, *production=True*)

Construct Grouper figshare groups

Parameters

- **group** (`Union[str, int]`) – Group name
- **root_stem** (`str`) – Grouper stem/folder for `group`
- **production** (`bool`) – Bool to use production stem. Otherwise a stage/test is used. Default: `True`

Return type `str`

Returns Grouper group string

Usage:

For active group, call as: figshare_group('active', '') > "arizona.edu:dept:LBRY:figshare:active"

For a quota group, call as: figshare_group('2147483648', 'quota') > "arizona.edu:dept:LBRY:figshare:quota:2147483648"

For a portal group, call as: figshare_group('sci_math', 'portal') > "arizona.edu:dept:LBRY:figshare:portal:sci_math"

requiam.commons.**figshare_stem**(*stem=''*, *production=True*)

Construct Grouper figshare stems

Parameters

- **stem** (str) – string corresponding to the sub-stem. Options are: 'quota', 'portal'. Default: root stem
- **production** (bool) – Bool to use production stem. Otherwise a stage/test is used. Default: True

Return type str

Returns Grouper stem/folder string

Usage:

For quota stem, call as: figshare_stem('quota') > "arizona.edu:dept:LBRY:figshare:quota"

For portal stem, call as: figshare_stem('portal') > "arizona.edu:dept:LBRY:figshare:portal"

For main stem, call as: figshare_stem() > "arizona.edu:dept:LBRY:figshare"

requiam.commons.**get_summary_dict**(*ldap_members*, *grouper_members*, *delta*)

Return a dict containing summary data for EDS and Grouper queries

Parameters

- **ldap_members** (set) – set containing EDS entries
- **grouper_members** (set) – set containing Grouper entries
- **delta** (*Delta*) – Delta object containing computation of adds and drops

Return type Dict[str, int]

Returns Python dict of containing summary data

requiam.commons.**int_conversion**(*string*)

Check and convert string that can be represented as an integer

Parameters **string** (str) – Input string

Return type Union[int, str]

Returns Result of conversion

delta module

class requiam.delta.**Delta**(*ldap_members*, *grouper_query_dict*, *batch_size*, *batch_timeout*, *batch_delay*, *sync_max*, *log=None*)

Bases: object

This class compares results from an LDAP query and a Grouper query to identify common, additions, and deletions so that the two will be in sync.

This code was adapted from the following repository: https://github.com/ualibraries/patron-groups

Usage: from requiam import delta

Parameters

- **ldap_members** (set) – Set of LDAP member ID
- **grouper_query_dict** (Dict[str, Any]) – Result from Grouper
- **batch_size** (int) – Number of records to synchronization for each "batch"
- **batch_timeout** (int) – Timeout in seconds for each batch
- **batch_delay** (int) – Delay between batches in seconds

- **sync_max** (int) – Maximum total adds and drops for synchronization
- **log** (Optional[Logger]) – Logger object

Variables

- **ldap_members** – Set of LDAP member IDs
- **grouper_query_dict** – Result from Grouper
- **grouper_members** – Set of Grouper member IDs
- **batch_size** – Number of records to synchronization for each "batch"
- **batch_timeout** – Timeout in seconds for each batch
- **batch_delay** – Delay between batches in seconds
- **sync_max** – Maximum total adds and drops for synchronization
- **log** – Logger object
- **adds** – Set of members to add to Grouper group
- **drops** – Set of members to drop from Grouper group
- **common** – Set of members in common with EDS/LDAP and Grouper

synchronize()

Return type None

grouper module

class requiam.grouper.**Grouper**(*grouper_host*, *grouper_base_path*, *grouper_user*, *grouper_password*, *grouper_production=False*, *log=None*)

Bases: object

This class uses the Grouper API to retrieve and send metadata

See Main Grouper API documentation.

Parameters

- **grouper_host** (str) – Grouper hostname (e.g., grouper.iam.arizona.edu)
- **grouper_base_path** (str) – Grouper base path that includes the API version (e.g., grouper-ws/servicesRest/json/v2_2_001)
- **grouper_user** (str) – Grouper username
- **grouper_password** (str) – Grouper password credential
- **grouper_production** (bool) – Bool to use production stem, figshare. Otherwise stage stem is used, figtest. Default: production

Variables

- **grouper_host** – Grouper hostname
- **grouper_base_path** – Grouper base path that includes the API version
- **grouper_user** – Grouper username
- **grouper_password** – Grouper password credential

- **grouper_production** – Bool to use production stem, `figshare`. Otherwise stage stem is used, `figtest`
- **grouper_auth** (*tuple*) – Grouper credential
- **endpoint** (*str*) – Grouper endpoint
- **headers** (*dict*) – HTTPS header information

add_group(*group*, *group_type*, *description*)

Create Grouper group within a Grouper stem

See Grouper API "Group Save"

Parameters

- **group** (`str`) – Grouper full group path from *requiam.commons.figshare_group()*
- **group_type** (`str`) – Grouper stem from *requiam.commons.figshare_stem()*. Options are: 'portal', 'quota', 'test', 'group_active', ''
- **description** (`str`) – Description of group to include as metadata. This shows up in the Grouper UI

Raises

- **ValueError** – If incorrect `group_type`
- **HTTPError** – If the Grouper POST fails with a non-200 status

Return type `bool`

add_privilege(*access_group*, *target_group*, *target_group_type*, *privileges*)

Add privilege(s) for a Grouper group to access target

See Grouper API "Add or remove Grouper privileges"

Parameters

- **access_group** (`str`) – Grouper group to give access to, ex: arizona.edu:Dept:LBRY:figshare:GrouperSuperAdmins
- **target_group** (`str`) – Grouper group to add privilege on, ex: "apitest"
- **target_group_type** (`str`) – Grouper stem associated with the group to add privilege on, ex: use 'figtest' for 'arizona.edu:Dept:LBRY:figtest:test'
- **privileges** (`Union[str, List[str]]`) – Grouper privileges. Allowed values: 'read', 'view', 'update', 'admin', 'optin', 'optout'

Raises

- **ValueError** – Incorrect `privileges` or Grouper POST failed
- **KeyError** – Incorrect `target_group_type`
- **Exception** – Incorrect `access_group` (check for existence)

Return type `bool`

Returns True on success, otherwise raises an Exception

check_group_exists(*group*, *group_type*)

Check whether a Grouper group exists within a Grouper stem

See Grouper API "Find Groups"

Parameters

- **group** (str) – Grouper full group path from *requiam.commons.figshare_group()*
- **group_type** (str) – Grouper stem. Options are: 'portal', 'quota', 'test', 'group_active', ''

Raises

- **ValueError** – If incorrect group_type
- **KeyError** – Stem does not exists

Return type bool

get_group_details(*group*)

Retrieve group details

See Grouper API "Get Groups" but using WsRestFindGroupsRequest

Parameters **group** (str) – Grouper path from *requiam.commons.figshare_group()*

Return type Any

Returns JSON response

get_group_list(*group_type*)

Retrieve list of groups in a Grouper stem

See Grouper API "Get Groups" but with a different implementation using FIND_BY_STEM_NAME method

Parameters **group_type** (str) – Grouper stem. Options are: 'portal', 'quota', 'test', 'group_active', ''. Note: Some groups (e.g., 'group_active') do not exist for production

Raises **ValueError** – If incorrect group_type

Return type Any

Returns JSON response

query(*group*)

Query Grouper for list of members in a group.

Parameters **group** (str) – Grouper full group path from *requiam.commons.figshare_group()*

Return type Dict[str, Any]

Returns Grouper metadata

url(*endpoint*)

Return full Grouper URL endpoint

Parameters **endpoint** (str) – The URL endpoint to append to self.endpoint

Return type str

Returns Complete HTTPS URL

requiam.grouper.**create_active_group**(*group*, *grouper_dict*, *group_description=None*, *log=None*, *add=False*)

Create a temporary group for figshare:active indirect membership

Parameters

- **group** (str) – Name of group (e.g., "ual")
- **grouper_dict** (dict) – Grouper configuration settings
- **group_description** (Optional[str]) – Grouper description. Defaults will prompt for it

- **log** (Optional[Logger]) – Logging object
- **add** (bool) – Indicate adding group. Default: False (dry run)

Return type None

requiam.grouper.**create_groups**(*groups*, *group_type*, *group_descriptions*, *grouper_api*, *log0=None*, *add=False*)

Process through a list of Grouper groups and add them if they don't exist and set permissions

Parameters

- **groups** (Union[str, List[str]]) – List containing group names
- **group_type** (str) – Grouper stem name. Either 'portal', 'quota', or 'test'
- **group_descriptions** (Union[str, List[str]]) – Descriptions of group to include as metadata. This shows up in the Grouper UI
- **grouper_api** (*Grouper*) – Grouper object
- **log0** (Optional[Logger]) – Logging object
- **add** (bool) – Indicate whether to perform update or dry run. Default: False

Raises **HTTPError** – Grouper POST fails

Return type None

requiam.grouper.**grouper_delta_user**(*group*, *stem*, *netid*, *uaid*, *action*, *grouper_dict*, *delta_dict*, *mo=None*, *sync=False*, *log=None*, *production=True*)

Construct a Delta object for addition/deletion based for a specified user. This is designed primarily for the user_update script

Parameters

- **group** (str) – The Grouper group to update
- **stem** (str) – The Grouper stem (e.g., 'portal', 'quota')
- **netid** (Union[str, List[str]]) – User NetID(s)
- **uaid** (Union[str, List[str]]) – User UA ID(s)
- **action** (str) – Action to perform. 'add' or 'remove'
- **grouper_dict** (Dict[str, Any]) – *requiam.grouper.Grouper* settings
- **delta_dict** (Dict[str, Any]) – *requiam.delta.Delta* settings
- **mo** (Optional[*ManualOverride*]) – *requiam.manual_override.ManualOverride* object Default: None
- **sync** (bool) – Indicate whether to sync. Default: False
- **log** (Optional[Logger]) – LogClass object. Default: None
- **production** (bool) – Use production stem. Otherwise a stage/test is used. Default: True

Return type *Delta*

Returns Delta object

ldap_query module

class requiam.ldap_query.**LDAPConnection**(*ldap_host*, *ldap_base_dn*, *ldap_user*, *ldap_password*, *log=<Logger stdout_logger (INFO)>*)

Bases: object

This class initializes a connection to a specified LDAP/EDS server. It allows for repeated LDAP queries. Originally patron group developed the connection to use with individual queries. The queries have been broken off since our use with the data repository could involve up to 1000 queries given the number of different organizations that we have.

Usage:

```
from requiam import ldap_query
eds_hostname = 'eds.arizona.edu'
ldap_base_dn = 'dc=eds,dc=arizona,dc=edu'
ldc = ldap_query.LDAPConnection(eds_hostname, ldap_base_dn,
                                USERNAME, PASSWORD)

portal_query = ldap_query.ual_ldap_queries(['0404', '0413', '0411'])
members = ldap_query.ldap_search(ldc, portal_query)
```

Parameters

- **ldap_host** (str) – LDAP host URL
- **ldap_base_dn** (str) – LDAP base distinguished name
- **ldap_user** (str) – LDAP username
- **ldap_password** (str) – LDAP password credentials
- **log** (Logger) – File and/or stdout logging. Default: log_stdout

Variables

- **ldap_host** – LDAP host URL
- **ldap_base_dn** – LDAP base distinguished name
- **ldap_user** – LDAP username
- **ldap_password** – LDAP password credentials
- **log** – File and/or stdout logging
- **ldap_bind_host** (*str*) – LDAP binding host URL
- **ldap_bind_dn** (*str*) – LDAP binding distinguished name
- **ldap_search_dn** (*str*) – LDAP search distinguished name
- **ldap_attribs** (*list*) – LDAP attributes. Set to "uaid"

requiam.ldap_query.**ldap_search**(*ldapconnection*, *ldap_query*)

Queries a define LDAP connection and retrieve members

Usage (see description in *requiam.ldap_query.LDAPConnection*):

```
members = ldap_query.ldap_search(ldc, ldap_query)
```

Parameters

- **ldapconnection** (*LDAPConnection*) – An ldap3 Connection from *requiam.ldap_query.LDAPConnection*
- **ldap_query** (list) – List of strings from *requiam.ldap_query.ual_ldap_queries()*

Return type set

Returns List of members

requiam.ldap_query.**ual_grouper_base**(*basename*)

Returns a string to use in LDAP queries that provide the Grouper ismemberof stem organization that UA Libraries use for patron management

Note that this only returns a string, it is not RFC 4512 compatible. See *requiam.ldap_query.ual_ldap_query()*

Usage:

```
grouper_base = ldap_query.ual_grouper_base('ual-faculty')
> "ismemberof=arizona.edu:dept:LBRY:pgrps:ual-faculty"
```

Parameters **basename** (str) – Grouper group name basename. Options are: ual-dcc, ual-faculty, ual-hsl, ual-staff, ual-students, ual-grads, ual-ugrads

Return type str

Returns ismemberof attribute

requiam.ldap_query.**ual_ldap_queries**(*org_codes*)

Construct *multiple* RFC 4512-compatible LDAP queries to search for those with UArizona Library privileges within multiple organizations specified by the org_codes input

Usage:

```
ldap_queries = ldap_query.ual_ldap_queries(['0212','0213','0214'])
```

Parameters **org_codes** (List[str]) – Organizational codes

Return ldap_queries list of str

Return type list

requiam.ldap_query.**ual_ldap_query**(*org_code*, *classification='all'*)

Construct RFC 4512-compatible LDAP query to search for those with UArizona Library privileges within an organization (specified by org_code)

Usage:

```
ldap_query = ldap_query.ual_ldap_query('0212')
> ['(& (employeePrimaryDept=0212) (|
    (ismemberof=arizona.edu:dept:LBRY:pgrps:ual-faculty)
    (ismemberof=arizona.edu:dept:LBRY:pgrps:ual-staff)
    (ismemberof=arizona.edu:dept:LBRY:pgrps:ual-students)
    (ismemberof=arizona.edu:dept:LBRY:pgrps:ual-dcc) ) )']
```

Parameters

- **org_code** (str) – Organizational code (e.g., '0212')

- **classification** (str) – Input for classification. Default: 'all'. Others: 'faculty', 'staff', 'students', 'dcc', 'none'. The 'none' input will provide an org_code-only query

Return type list

Returns LDAP query

requiam.ldap_query.**uid_query**(*uid*)

Construct RFC 4512-compatible LDAP query for a single NetID account

Usage:

```
ldap_query = ldap_query.ual_test_query('<netid>')
> ['(uid=<netid>)']
```

Parameters **uid** (str) – NetID handle/username

Return type list

Returns LDAP query

manual_override module

class requiam.manual_override.**ManualOverride**(*portal_file*, *quota_file*, *log=<Logger stdout_logger (INFO)>*, *root_add=False*)

Bases: object

This class handles manual override changes. It reads in CSV configuration files and queries pandas.DataFrame to identify additions/deletions. It employ set operations for simplicity. It also update the CSV files after a change is implemented

Parameters

- **portal_file** (str) – Full file path for CSV file containing manual portal specifications (e.g., config/portal_manual.csv)
- **quota_file** (str) – Full file path for CSV file containing manual quota specifications (e.g., config/quota_manual.csv)
- **log** (Logger) – File and/or stdout logging
- **root_add** (bool) – Flag to set root as portal in manual CSV file. Default: False. In the default case, a force to "root" will delete existing records in the manual quota CSV. If user ID is not present, nothing happens

Variables

- **portal_file** (*str*) – Full file path for CSV file containing manual portal specification
- **quota_file** (*str*) – Full file path for CSV file containing manual quota specification
- **log** (*Logger*) – File and/or stdout logging
- **portal_df** (*pd.DataFrame*) – Portal DataFrame
- **quota_df** (*pd.DataFrame*) – Quota DataFrame
- **portal_header** (*list*) – CSV header for portal_df
- **quota_header** (*list*) – CSV header for quota_df

file_checks(*input_file*)

Checks to see if manual CSV file exists.

Parameters **input_file** (str) – Path of file to check

Return type bool

Returns Result of file check

identify_changes(*ldap_set*, *group*, *group_type*)

Identify changes to call *requiam.manual_override.update_entries()* accordingly

Parameters

- **ldap_set** (set) – Input EDS user IDs
- **group** (str) – Group to identify membership
- **group_type** (str) – Manual CSV type. Either 'portal' or 'quota'

Raises **ValueError** – Incorrect input on group_type

Return type set

Returns EDS user IDs with changes (after addition and deletion)

read_manual_file(*group_type*)

Return a pandas.DataFrame containing the manual override file

Parameters **group_type** (str) – Grouper group type. Either 'portal' or 'quota'

Raises

- **ValueError** – Incorrect input on group_type
- **FileNotFound** – Unable to find manual CSV to load

Return type DataFrame

Returns DataFrame corresponding to group_type

update_dataframe(*netid*, *uaid*, *group*, *group_type*)

Update pandas.DataFrame with necessary changes

Parameters

- **netid** (list) – UA NetIDs
- **uaid** (list) – UA IDs
- **group** (str) – Group to identify membership
- **group_type** (str) – Manual CSV type. Either 'portal' or 'quota'

Raises **ValueError** – Incorrect input on group_type

Return type None

requiam.manual_override.**csv_commented_header**(*input_file*)

Read in the comment header in CSV file to re-populate later

Parameters **input_file** (str) – Full path to CSV file

Return type list

Returns CSV header

requiam.manual_override.**get_current_groups**(*uid*, *ldap_dict*, *production=False*, *log=<Logger stdout_logger (INFO)>*, *verbose=True*)

Retrieve current Figshare ismemberof association

Parameters

- **uid** (str) – User NetID
- **ldap_dict** (dict) – LDAP settings
- **production** (bool) – Flag to indicate using Grouper production stem (figshare) over test (figtest). Default: False
- **log** (Logger) – File and/or stdout logging
- **verbose** (bool) – Provide information about each user. Default: True

Raises **ValueError** – User is associated with multiple portal/quota groups

Return figshare_dict dict containing current Figshare portal and quota

Return type dict

requiam.manual_override.**update_entries**(*ldap_set*, *netid*, *uaid*, *action*, *log=<Logger stdout_logger (INFO)>*)

Add/remove entries from a set

Parameters

- **ldap_set** (set) – UA IDs from EDS
- **netid** (list) – UA NetIDs to add/remove
- **uaid** (list) – UA IDs for corresponding netid
- **action** (str) – Action to perform. Either 'remove' or 'add'
- **log** (Logger) – File and/or stdout Logger object

Raises **ValueError** – Incorrect action setting

Return type set

Returns Updated set of uaid values

org_code_numbers module

requiam.org_code_numbers.**get_numbers**(*lc*, *org_url*, *log*)

Determine number of individuals in each organization code with Library privileges and write to a file called "org_code_numbers.csv"

Parameters

- **lc** (*LDAPConnection*) – LDAPConnection object for EDS record retrieval
- **org_url** (str) – Google Docs URL that provides CSV
- **log** (Logger) – File and/or stdout logging class

Raises **URLError** – Incorrect org_url

Return type None

quota module

requiam.quota.**ual_ldap_quota_query**(*ual_class*, *org_codes=None*)

Construct RFC 4512-compatible LDAP query to search for those within a UAL-based classification patron group

This function provides LDAP information for IAM accounts associated with default quota tiers (faculty, grad, undergrad)

It is intended to be used with the *requiam.ldap_query.LDAPConnection* object through *requiam.ldap_query.ldap_search()*:

```
quota_query = ual_ldap_quota_query('faculty')
members     = ldap_query.ldap_search(ldc, quota_query)
```

Parameters

- **ual_class** (str) – UA classification. Options are:
  - "faculty" (for faculty, staff, and DCCs)
  - "grad" (for graduate students)
  - "ugrad" (for undergraduate students)
- **org_codes** (Optional[list]) – Org codes to require in search.

Raises **SystemExit** – Incorrect ual_class input

Return type Optional[list]

Returns List containing query/queries

9.1.2 Additional Classes

class requiam.**TimerClass**

Bases: object

Define timer object that records elapsed time

Usage:

```
# Initiate
timer = TimerClass()
timer._start()

# Stop
timer._stop()

# Get information
timer.format
```

Variables

- **start** – Starting time
- **stop** – Stopping time
- **delta** – Difference between start and stop
- **format** (*str*) – Duration in human readable form

CHAPTER

TEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

r

requiam.commons, 19
requiam.delta, 20
requiam.grouper, 21
requiam.ldap_query, 25
requiam.manual_override, 27
requiam.org_code_numbers, 29
requiam.quota, 30

INDEX

A

add_group() (*requiam.grouper.Grouper method*), 22
add_privilege() (*requiam.grouper.Grouper method*), 22

C

check_group_exists() (*requiam.grouper.Grouper method*), 22
create_active_group() (*in module requiam.grouper*), 23
create_groups() (*in module requiam.grouper*), 24
csv_commented_header() (*in module requiam.manual_override*), 28

D

Delta (*class in requiam.delta*), 20
dict_load() (*in module requiam.commons*), 19

F

figshare_group() (*in module requiam.commons*), 19
figshare_stem() (*in module requiam.commons*), 19
file_checks() (*requiam.manual_override.ManualOverride method*), 27

G

get_current_groups() (*in module requiam.manual_override*), 28
get_group_details() (*requiam.grouper.Grouper method*), 23
get_group_list() (*requiam.grouper.Grouper method*), 23
get_numbers() (*in module requiam.org_code_numbers*), 29
get_summary_dict() (*in module requiam.commons*), 20
Grouper (*class in requiam.grouper*), 21
grouper_delta_user() (*in module requiam.grouper*), 24

I

identify_changes() (*requiam.manual_override.ManualOverride method*), 28
int_conversion() (*in module requiam.commons*), 20

L

ldap_search() (*in module requiam.ldap_query*), 25
LDAPConnection (*class in requiam.ldap_query*), 25

M

ManualOverride (*class in requiam.manual_override*), 27
module
    requiam.commons, 19
    requiam.delta, 20
    requiam.grouper, 21
    requiam.ldap_query, 25
    requiam.manual_override, 27
    requiam.org_code_numbers, 29
    requiam.quota, 30

Q

query() (*requiam.grouper.Grouper method*), 23

R

read_manual_file() (*requiam.manual_override.ManualOverride method*), 28
requiam.commons
    module, 19
requiam.delta
    module, 20
requiam.grouper
    module, 21
requiam.ldap_query
    module, 25
requiam.manual_override
    module, 27
requiam.org_code_numbers
    module, 29
requiam.quota
    module, 30

S

`synchronize()` (*requiam.delta.Delta method*), 21

T

`TimerClass` (*class in requiam*), 30

U

`ual_grouper_base()` (*in module requiam.ldap_query*), 26
`ual_ldap_queries()` (*in module requiam.ldap_query*), 26
`ual_ldap_query()` (*in module requiam.ldap_query*), 26
`ual_ldap_quota_query()` (*in module requiam.quota*), 30
`uid_query()` (*in module requiam.ldap_query*), 27
`update_dataframe()` (*requiam.manual_override.ManualOverride method*), 28
`update_entries()` (*in module requiam.manual_override*), 29
`url()` (*requiam.grouper.Grouper method*), 23