
Pacifica Policy Documentation

David Brown

Jul 11, 2019

Contents:

1	Installation	3
1.1	Installation in Virtual Environment	3
2	Configuration	5
2.1	CherryPy Configuration File	5
2.2	Service Configuration File	5
2.3	Starting the Service	6
3	Example Usage	9
3.1	The API	9
3.2	Admin Command Line	11
4	Policy Python Module	13
4.1	Events Python Module	13
4.2	Ingest Python Module	13
4.3	Reporting Python Module	14
4.4	Status Python Module	15
4.5	Uploader Python Module	18
4.6	Admin Python Module	19
4.7	Admin Command Python Module	20
4.8	Config Python Module	20
4.9	Data Release Python Module	20
4.10	Globals Python Module	20
4.11	Root Rest Python Module	20
4.12	Search Render Python Module	21
4.13	Search Sync Python Module	21
4.14	Validation Python Module	21
4.15	WSGI Python Module	22
5	Indices and tables	23
	Python Module Index	25
	Index	27

The Pacifica Policy service provides endpoints that define policy questions for institutions. This is separate from other services as certain operations required by other Pacifica Core services are more Policy base.

Practially speaking, when the question a Pacifica service wants to ask the Metadata service is sufficiently complex it should really be a Policy question. For example, when uploading data the ingest service needs to validate the metadata requesting to be added. This new metadata needs to be verified by some institutional requirements. So there is a Policy endpoint (several actually) that help ensure those requirements are met.

The Pacifica software is available through PyPi so creating a virtual environment to install is what is shown below. Please keep in mind compatibility with the Pacifica Core services.

1.1 Installation in Virtual Environment

These installation instructions are intended to work on both Windows, Linux, and Mac platforms. Please keep that in mind when following the instructions.

Please install the appropriate tested version of Python for maximum chance of success.

1.1.1 Linux and Mac Installation

```
mkdir ~/.virtualenvs
python -m virtualenv ~/.virtualenvs/pacifica
. ~/.virtualenvs/pacifica/bin/activate
pip install pacifica-policy
```

1.1.2 Windows Installation

This is done using PowerShell. Please do not use Batch Command.

```
mkdir "$Env:LOCALAPPDATA\virtualenvs"
python.exe -m virtualenv "$Env:LOCALAPPDATA\virtualenvs\pacifica"
& "$Env:LOCALAPPDATA\virtualenvs\pacifica\Scripts\activate.ps1"
pip install pacifica-policy
```


The Pacifica Core services require two configuration files. The REST API utilizes [CherryPy](#) and review of their [configuration documentation](#) is recommended. The service configuration file is a INI formatted file containing configuration for database connections.

2.1 CherryPy Configuration File

An example of Policy server CherryPy configuration:

```
[global]
log.screen: True
log.access_file: 'access.log'
log.error_file: 'error.log'
server.socket_host: '0.0.0.0'
server.socket_port: 8181

[/]
request.dispatch: cherrypy.dispatch.MethodDispatcher()
tools.response_headers.on: True
tools.response_headers.headers: [('Content-Type', 'application/json')]
```

2.2 Service Configuration File

The service configuration is an INI file and an example is as follows:

```
[policy]
; This section has policy service specific config options

; The following strings reference formatting directives {}. The
; object passed to the format method is the transaction object
```

(continues on next page)

```
; from the metadata API. The DOI is special and added into the
; transaction object for that format as well.

; Internal URL format for transactions not released or have DOIs
internal_url_format = https://internal.example.com/{_id}

; Release URL format for transactions released but no DOI
release_url_format = https://release.example.com/{_id}

; DOI URL format for transactions with a DOI
doi_url_format = https://dx.doi.org/{doi}

; In memory object cache size (used in data release)
cache_size = 10000

; This sets the admin group name
admin_group = admin

; This sets the admin group id (should match group name in metadata)
admin_group_id = 0

; This sets the admin user id (should match user name in metadata)
admin_user_id = 0

[metadata]
; This section contains configuration for metadata service

; The global metadata url
endpoint_url = http://localhost:8121

; The endpoint to check for status of metadata service
status_url = http://localhost:8121/groups

[elasticsearch]
; This section describes configuration to contact elasticsearch

; URL to the elasticsearch server
url = http://127.0.0.1:9200

; URL to the elasticsearch server
index = pacifica_search

; Timeout for connecting to elasticsearch
timeout = 60

; Turn on or off elasticsearch sniffing
; https://elasticsearch-py.readthedocs.io/en/master/#sniffing
sniff = True
```

2.3 Starting the Service

Starting the Policy service can be done by two methods. However, understanding the requirements and how they apply to REST services is important to address as well. Using the internal CherryPy server to start the service is recommended for Windows platforms. For Linux/Mac platforms it is recommended to deploy the service with uWSGI.

2.3.1 Deployment Considerations

The Policy server can have the same memory consumption issues as the Metadata service. Please consider those recommendations [here](#) similarly for the Policy service.

2.3.2 CherryPy Server

To make running the Policy service using the CherryPy's builtin server easier we have a command line entry point.

```
$ pacifica-policy --help
usage: pacifica-policy [-h] [-c CONFIG] [-p PORT] [-a ADDRESS]

Run the policy server.

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG, --config CONFIG
                        cherrypy config file
  -p PORT, --port PORT  port to listen on
  -a ADDRESS, --address ADDRESS
                        address to listen on

$ pacifica-policy
[09/Jan/2019:09:17:26] ENGINE Listening for SIGTERM.
[09/Jan/2019:09:17:26] ENGINE Bus STARTING
[09/Jan/2019:09:17:26] ENGINE Set handler for console events.
[09/Jan/2019:09:17:26] ENGINE Started monitor thread 'Autoreloader'.
[09/Jan/2019:09:17:26] ENGINE Serving on http://0.0.0.0:8181
[09/Jan/2019:09:17:26] ENGINE Bus STARTED
```

2.3.3 uWSGI Server

To make running the Policy service using uWSGI easier we have a module to be included as part of the uWSGI configuration. uWSGI is very configurable and can use this module many different ways. Please consult the [uWSGI Configuration](#) documentation for more complicated deployments.

```
$ pip install uwsgi
$ uwsgi --http-socket :8181 --master --module pacifica.policy.wsgi
```

Example Usage

The usage of the Policy API is strictly a read-only interface. The command line usage of the system provides tools to update data in the metadata API and are mostly cronjob like processes.

3.1 The API

The policy server is split up into endpoints named for their Pacifica project that utilizes them. So the path `/uploader` is used by the Pacifica Uploader (<http://github.com/pacifica/pacifica-uploader>) to control its behavior. The idea is that workflow implemented by the various Pacifica projects has some element of site or instance specific policy that can be applied to the running service. The policy is driven by the metadata and thus this project should talk to the metadata service.

3.1.1 Events API

The Events API is used by the [Notifications](#) service. The role of this query is to verify the event recieved by the Notifications services is allowed to be sent to the user on the URL path.

Request Example:

```
POST /events/dmlb2001
Content-Type: application/json
{
  "data": [
    ...
  ]
}
```

Good Response Example:

```
Http-Code: 200
{
```

(continues on next page)

(continued from previous page)

```
"status": "success"
}
```

Failed Response Example:

```
Http-Code: 401
{
  "error": "...
}
```

The underlying logic for this implementation is the same as the ingest endpoint discussed next.

3.1.2 Ingest API

The Ingest API is used by the [Ingest](#) service. This endpoint verifies the relationships between user, project and instrument before allowing an upload. The content of the body document is defined by the [uploader](#).

Request Example:

```
POST /ingest
Content-Type: application/json
[
  ...
]
```

Good Response Example:

```
Http-Code: 200
{
  "status": "success"
}
```

Failed Response Example:

```
Http-Code: 401
{
  "error": "...
}
```

3.1.3 Reporting and Status API

This document is not going into details about these APIs currently. These endpoints are supposed to be used by tools that provide status of current uploads to users of Pacifica as well as institutional reporting tools that aggregate metrics about uploads in Pacifica. Eventually, Pacifica should have a basic set of these websites to allow users to use these endpoints but not currently.

3.1.4 Uploader API

The Uploader API is a simple query interface to get complex metadata interactively while users are using the Uploader. This API has a JSON document that looks very SQL like but is not complete.

Request Example:

```

POST /uploader
Content-Type: application/json
{
  "user": 100,
  "from": "instruments",
  "columns": [
    "_id",
    "name"
  ],
  "where": {
    "_id": 54
  }
}

```

Good Response Example:

```

Http-Code: 200
[
  {
    "_id": 54,
    "name": "NMR PROBES: Nittany Liquid"
  }
]

```

Failed Response Example:

```

Http-Code: 500

```

3.2 Admin Command Line

There is a single admin command line tool (`pacifica-policy-cmd`) with two subcommands, `data_release` and `searchsync`. The `data_release` subcommand handles setting the `data_release` attributes of the Projects and Transactions. The `searchsync` subcommand handles formatting and synchronizing metadata to [ElasticSearch](#).

```

$ pacifica-policy-cmd --help
usage: pacifica-policy-cmd [-h] [--verbose] {data_release,searchsync} ...

positional arguments:
  {data_release,searchsync}
                        sub-command help
  data_release         data_release help
  searchsync          searchsync help

optional arguments:
  -h, --help           show this help message and exit
  --verbose            enable verbose debug output

```

3.2.1 Data Release

The data release process involves two phases, updating the suspense date and setting data release. The suspense date is a date that the metadata and data associated with that object in metadata will be released in the future. The data release phase checks the suspense date with now to determine if the object needs to have it released.

```
$ pacifica-policy-cmd data_release --help
usage: pacifica-policy-cmd data_release [-h]
                                     [--exclude [EXCLUDE [EXCLUDE ...]]]
                                     [--keyword KEYWORD]
                                     [--time-after TIME_AFTER]
                                     [--time-ago TIME_AGO]

data release by policy

optional arguments:
  -h, --help            show this help message and exit
  --exclude [EXCLUDE [EXCLUDE ...]]
                        id of keyword prefix to exclude.
  --keyword KEYWORD     keyword one of projects.actual_end_date,
                        projects.actual_start_date, projects.submitted_date,
                        projects.accepted_date, projects.closed_date,
                        transactions.created, transactions.updated.
  --time-after TIME_AFTER
                        set suspense date on data to X days after keyword.
  --time-ago TIME_AGO  only objects updated after X days ago.
```

Example command lines from the test suite.

```
pacifica-search-cmd data_release --time-after='365 days after' --exclude='1234cé'
pacifica-search-cmd data_release --keyword='transactions.created' --verbose
```

3.2.2 Search Sync

The search synchronization to Elasticsearch is driven by the Policy service. The metadata in Elasticsearch is meant to be consumed by client applications and in order to be performant those clients should communicate directly with Elasticsearch. This does mean that the metadata in Elasticsearch is not as current as the Metadata API.

```
$ pacifica-policy-cmd searchsync
usage: pacifica-policy-cmd searchsync [-h] [--objects-per-page ITEMS_PER_PAGE]
                                     [--threads THREADS]
                                     [--time-ago TIME_AGO]

sync sql data to elastic for search

optional arguments:
  -h, --help            show this help message and exit
  --objects-per-page ITEMS_PER_PAGE
                        objects per bulk upload.
  --threads THREADS     number of threads to sync data
  --time-ago TIME_AGO  only objects newer than X days ago.
```

Example command lines from the test suite.

```
pacifica-policy-cmd searchsync --objects-per-page=4 --threads=1 --time-ago='7 days ago'
↪ --exclude='keys._id=104'
```


4.1 Events Python Module

Events module to drive policy for who can see events.

Events rest module for the cherrypy endpoint.

```
class pacifica.policy.events.rest.EventsPolicy  
    CherryPy Events Policy.
```

This exposes whether a user can see an event from.

```
POST (username)
```

Pull the json content and validate the user can see the event.

```
exposed = True
```

4.2 Ingest Python Module

Ingest valication module.

The CherryPy rest object for the structure.

Below is an example post body:

```
[  
  {"destinationTable": "Transactions._id", "value": 1234},  
  {"destinationTable": "Transactions.submitter", "value": 34002},  
  {"destinationTable": "Transactions.project", "value": "34002"},  
  {"destinationTable": "Transactions.instrument", "value": 34002},  
  {"destinationTable": "TransactionKeyValue", "key": "Tag", "value": "Blah"},  
  {"destinationTable": "TransactionKeyValue", "key": "Taggy", "value": "Blah"},  
  {"destinationTable": "TransactionKeyValue", "key": "Taggier", "value": "Blah"}  
]
```

(continues on next page)

```

    "destinationTable": "Files",
    "_id": 34, "name": "foo.txt", "subdir": "a/b/",
    "ctime": "Tue Nov 29 14:09:05 PST 2016",
    "mtime": "Tue Nov 29 14:09:05 PST 2016",
    "size": 128, "mimetype": "text/plain"
  },
  {
    "destinationTable": "Files",
    "_id": 35, "name": "bar.txt", "subdir": "a/b/",
    "ctime": "Tue Nov 29 14:09:05 PST 2016",
    "mtime": "Tue Nov 29 14:09:05 PST 2016",
    "size": 47, "mimetype": "text/plain"
  },
]

```

class pacifica.policy.ingest.rest.IngestPolicy

CherryPy Ingest Policy Class.

POST ()

Read in the json query and return results.

static **_pull_data_by_rec** (*query, table*)

Pull the value for the table.

_valid_query (*query*)

Validate the metadata format.

4.3 Reporting Python Module

CherryPy Uploader Policy object class.

CherryPy Uploader Policy object class.

CherryPy Status Metadata projectinfo base class.

class pacifica.policy.reporting.transaction.query_base.QueryBase

Formats summary data for other classes down the tree.

static **_get_user_lookups** (*url, header_list*)

static **_merge_two_dicts** (*dict_a, dict_b*)

Given two dicts, merge them into a new dict as a shallow copy.

base_user_info = {'emsl_employee': False, 'instrument_list': [], 'project_list': []}

static **get_full_user_info** (*user_id*)

Return user information for the given user_id.

CherryPy Status Metadata object class.

class pacifica.policy.reporting.transaction.transaction_details.TransactionDetails

Retrieves a list of all transactions matching the search criteria.

static **POST** (*user_id=None*)

CherryPy GET method.

static **_get_transaction_list_details** (*transaction_list, user_id*)

exposed = True

CherryPy Status Metadata object class.

```
class pacifica.policy.reporting.transaction.transaction_summary.TransactionSummary
    Retrieves a summary of all transactions matching the search criteria.

    static POST (time_basis=None, object_type=None, start_date=None, end_date=None, **kwargs)
        CherryPy GET method.

    static _cleanup_object_stats (object_listing, object_type, user_info)

    static _get_transaction_list_summary (time_basis, object_list, object_type, start_date,
                                         end_date, user_id)

    exposed = True
```

The CherryPy rest object for the structure.

```
class pacifica.policy.reporting.rest.ReportingPolicy
    CherryPy root object class.

    not exposed by default the base objects are exposed.

    __init__ ()
        Create local objects to allow for import to work.

    exposed = False
```

4.4 Status Python Module

CherryPy Uploader Policy object class.

Base class module for standard queries for the upload status tool.

```
class pacifica.policy.status.base.QueryBase
    This pulls the common bits of instrument and project query into a single class.

    _get_available_projects (user_id)

    all_instruments_url = 'http://localhost:8121/instruments'
    all_projects_url = 'http://localhost:8121/projects'
    all_transactions_url = 'http://localhost:8121/transactions'
    md_url = 'http://localhost:8121'
```

CherryPy Status Policy object class.

```
class pacifica.policy.status.instrument_query.InstrumentQuery
    CherryPy root object class.

    __init__ ()
        Create local objects for sub tree items.

    exposed = False
```

CherryPy Status Policy object class.

```
class pacifica.policy.status.project_query.ProjectQuery
    CherryPy root object class.

    __init__ ()
        Create local objects for sub tree items.

    exposed = False
```

The CherryPy rest object for the structure.

```
class pacifica.policy.status.rest.StatusPolicy
    CherryPy root object class.

    not exposed by default the base objects are exposed.

    __init__()
        Create local objects to allow for import to work.

    exposed = False
```

CherryPy Status Policy object class.

```
class pacifica.policy.status.transaction_query.TransactionQuery
    CherryPy root object class.

    __init__()
        Create local objects for sub tree items.

    exposed = False
```

CherryPy Status Policy object class.

```
class pacifica.policy.status.user_query.UserQuery
    CherryPy root object class.

    __init__()
        Create local objects for sub tree items.

    exposed = True
```

CherryPy Project Policy object classes.

CherryPy Status Policy object class.

```
class pacifica.policy.status.instrument.by_project_id.InstrumentsByProject
    Retrieves instrument list for a given project.

    static GET (project_id=None)
        CherryPy GET method.

    static _get_instruments_for_project (project_id)
        Return a list with all the instruments belonging to this project.

    exposed = True
```

CherryPy Status Policy object class.

```
class pacifica.policy.status.instrument.search.InstrumentKeywordSearch
    Retrieves a set of projects for a given keyword set.

    GET (search_terms="", **kwargs)
        CherryPy GET method.

    _clean_up_instrument_list (inst_response, user_id)
        Clear out entries that done belong to this user.

    _get_instruments_for_keywords (user_id, search_terms="")
        Return a list with all the instruments having this term.

    static _squash_output_list (inst_for_user_list, full_inst_list)
        Filter entries in the full instrument list.

    exposed = True
```

CherryPy Project Policy object classes.

CherryPy Status Policy object class.

```
class pacifica.policy.status.project.by_user.ProjectUserSearch
    Retrieves project list for a given user.

    static GET (user_id=None)
        CherryPy GET method.

    static _get_projects_for_user (user_id=None)
        Return a list with all the projects involving this user.

    exposed = True
```

CherryPy Status Policy object class.

```
class pacifica.policy.status.project.lookup.ProjectLookup
    Retrieves details of a given project.

    static GET (project_id=None)
        CherryPy GET method.

    static _get_projects_details (project_id=None)
        Return a details about this project.

    exposed = True
```

CherryPy Status Policy object class.

```
class pacifica.policy.status.project.search.ProjectKeywordSearch
    Retrieves a set of projects for a given keyword set.

    GET (search_terms=None, **kwargs)
        CherryPy GET method.

    _get_projects_for_keywords (user_id, search_terms=None)
        Return a list with all the projects involving this user.

    exposed = True
```

CherryPy Uploader Policy object class.

CherryPy Status Policy object class.

```
class pacifica.policy.status.transaction.files.FileLookup
    Retrieves files for a given transaction_id.

    static GET (transaction_id=None)
        CherryPy GET method.

    static _get_file_list (transaction_id=None)
        Return files for the specified transaction entry.

    exposed = True
```

CherryPy Status Policy object class.

```
class pacifica.policy.status.transaction.lookup.TransactionLookup
    Retrieves details of a given project.

    static GET (transaction_id=None)
        CherryPy GET method.

    static _get_transaction_details (transaction_id=None)
        Return details for the specified transaction entry.
```

exposed = True

CherryPy Status Policy object class.

class pacifica.policy.status.transaction.search.**TransactionSearch**

Retrieves a set of transactions for a given keyword set.

static GET (*option=None, **kwargs*)

CherryPy GET method.

static _get_transactions_for_keywords (*kwargs, option=None*)

Return a list with all the projects involving this user.

exposed = True

CherryPy Uploader Policy object class.

CherryPy Status Policy object class.

class pacifica.policy.status.user.lookup.**UserLookup**

Retrieves info for the specified user.

static GET (*user_id=None*)

CherryPy GET method.

static _get_user_info (*user_id*)

Return detailed info about a given user.

exposed = True

CherryPy Status Policy object class.

class pacifica.policy.status.user.search.**UserSearch**

Retrieves a set of projects for a given keyword set.

GET (*search_terms=None, option=None, **kwargs*)

CherryPy GET method.

static _get_users_for_keywords (*kwargs, search_terms=None, option=None*)

Return a list with all the projects involving this user.

exposed = True

4.5 Uploader Python Module

CherryPy Uploader Policy object class.

The CherryPy rest object for the structure.

class pacifica.policy.uploader.rest.**UploaderPolicy**

CherryPy root object class.

not exposed by default the base objects are exposed

POST ()

Read in the json query and return results.

static _clean_user_query_id (*query*)

determine the user_id for whatever is in the query.

static _filter_results (*results, *args*)

_query_select (*query*)

```

_query_select_admin(query)
_query_select_instrument_info(query)
_query_select_project_info(query)
_query_select_user_info(query)
_user_info_from_queries(user_queries)
static _valid_query(query)
exposed = True

```

4.6 Admin Python Module

The Admin module has logic about checking for admin group info.

```

class pacifica.policy.admin.AdminPolicy
    Enforces the admin policy.

    Base class for checking for admin group membership or not.

    __init__()
        Constructor for Uploader Policy.

    _all_instrument_info()

    _all_project_info()

    _format_url(url, **get_args)
        Append the recursion_depth parameter to the url.

    _groups_for_inst(inst_id)

    _instrument_info_from_ids(inst_list)

    _instruments_for_custodian(user_id)

    _instruments_for_group(group_id)

    _instruments_for_user(user_id)

    _instruments_for_user_proj(user_id, proj_id)

    _is_admin(user_id)

    static _object_id_valid(object_lookup_name, object_id)

    _project_info_from_ids(proj_list)

    _projects_for_custodian(user_id)

    _projects_for_inst(inst_id)

    _projects_for_user(user_id, relationship='member_of')

    _projects_for_user_inst(user_id, inst_id)

    _user_info_from_kwds(**kwds)

    _users_for_proj(proj_id)

    all_instruments_url = 'http://localhost:8121/instruments'
    all_projects_url = 'http://localhost:8121/projects'

```

```
all_relationships_url = 'http://localhost:8121/relationships'  
all_users_url = 'http://localhost:8121/users'  
get_relationship_info (**get_args)  
    Get a relationship by kwargs.  
inst_group_url = 'http://localhost:8121/instrument_group'  
inst_user_url = 'http://localhost:8121/instrument_user'  
md_url = 'http://localhost:8121'  
proj_instrument_url = 'http://localhost:8121/project_instrument'  
proj_user_url = 'http://localhost:8121/project_user'
```

4.7 Admin Command Python Module

4.8 Config Python Module

Configuration reading and validation module.

```
pacifica.policy.config.get_config  
    Return the ConfigParser object with defaults set.
```

Currently metadata API doesn't work with SQLite the queries are too complex and it only is supported with MySQL and PostgreSQL.

4.9 Data Release Python Module

4.10 Globals Python Module

Global static variables.

4.11 Root Rest Python Module

CherryPy root object class.

```
class pacifica.policy.root.Root  
    CherryPy root object class.  
  
    not exposed by default the base objects are exposed  
  
    static GET()  
        Return happy message about functioning service.  
  
    __init__()  
        Create the local objects we need.  
  
    exposed = True  
  
    classmethod try_meta_connect (attempts=0)  
        Try to connect to the metadata service see if its there.
```


`pacifica.policy.root.error_page_default(**kwargs)`
 The default error page should always enforce json.

4.12 Search Render Python Module

This is the render object for the search interface.

```
class pacifica.policy.search_render.SearchRender
  Search render class to contain methods.

  classmethod generate (obj_cls, objs, exclude)
    generate the institution object.

  static get_render_class (obj_cls)
    Get the render class dynamically.

  classmethod object_exclude (obj_cls, obj, exclude)
    Check if object is part of the exclude tuples.
```

4.13 Search Sync Python Module

Sync the database to elasticsearch index for use by Searching tools.

```
pacifica.policy.search_sync.create_worker_threads (threads, work_queue)
  Create the worker threads and return the list.

pacifica.policy.search_sync.es_client ()
  Get the elasticsearch client object.

pacifica.policy.search_sync.generate_work (items_per_page, work_queue, time_ago, exclude)
  Generate the work from the db and send it to the work queue.

pacifica.policy.search_sync.search_sync (args)
  Main search sync subcommand.

pacifica.policy.search_sync.start_work (work_queue)
  The main thread for the work.

pacifica.policy.search_sync.try_doing_work (cli, job)
  Try doing some work even if you fail.

pacifica.policy.search_sync.try_es_connect (attempts=0)
  Recursively try to connect to elasticsearch.

pacifica.policy.search_sync.yield_data (**kwargs)
  yield objects from obj for bulk ingest.
```

4.14 Validation Python Module

Validation methods for various objects.

```
pacifica.policy.validation._get_check_id (index, *args, **kwargs)
  Return the check ID in args or kwargs.
```

`pacifica.policy.validation.validate_project` (*index=0*)
Validate the project id.

`pacifica.policy.validation.validate_transaction` (*index=0*)
Validate the transaction id.

`pacifica.policy.validation.validate_universal` (*index, regex*)
Decorator generator to validate project field.

`pacifica.policy.validation.validate_user` (*index=0*)
Validate the user id.

4.15 WSGI Python Module

This is the main policy server script. This is the policy module.

CHAPTER 5

Indices and tables

- `genindex`
- `modindex`
- `search`

p

`pacifica.policy`, 22

`pacifica.policy.admin`, 19

`pacifica.policy.config`, 20

`pacifica.policy.events`, 13

`pacifica.policy.events.rest`, 13

`pacifica.policy.globals`, 20

`pacifica.policy.ingest`, 13

`pacifica.policy.ingest.rest`, 13

`pacifica.policy.reporting`, 14

`pacifica.policy.reporting.rest`, 15

`pacifica.policy.reporting.transaction`, 14

`pacifica.policy.reporting.transaction.query_base`, 14

`pacifica.policy.reporting.transaction.transaction_details`, 14

`pacifica.policy.reporting.transaction.transaction_summary`, 15

`pacifica.policy.root`, 20

`pacifica.policy.search_render`, 21

`pacifica.policy.search_sync`, 21

`pacifica.policy.status`, 15

`pacifica.policy.status.base`, 15

`pacifica.policy.status.instrument`, 16

`pacifica.policy.status.instrument.by_project_id`, 16

`pacifica.policy.status.instrument.search`, 16

`pacifica.policy.status.instrument_query`, 15

`pacifica.policy.status.project`, 17

`pacifica.policy.status.project.by_user`, 17

`pacifica.policy.status.project.lookup`, 17

`pacifica.policy.status.project.search`, 17

`pacifica.policy.status.project_query`, 15

`pacifica.policy.status.rest`, 16

`pacifica.policy.status.transaction`, 17

`pacifica.policy.status.transaction.files`, 17

`pacifica.policy.status.transaction.lookup`, 17

`pacifica.policy.status.transaction.search`, 18

`pacifica.policy.status.transaction_query`, 16

`pacifica.policy.status.user`, 18

`pacifica.policy.status.user.lookup`, 18

`pacifica.policy.status.user.search`, 18

`pacifica.policy.status.user_query`, 16

`pacifica.policy.uploader`, 18

`pacifica.policy.uploader.rest`, 18

`pacifica.policy.validation`, 21

`pacifica.policy.wsgi`, 22

Symbols

<code>__init__()</code> (<i>pacifica.policy.admin.AdminPolicy</i> method), 19	<code>_get_check_id()</code> (in module <i>pacifica.policy.validation</i>), 21
<code>__init__()</code> (<i>pacifica.policy.reporting.rest.ReportingPolicy</i> method), 15	<code>_get_file_list()</code> (<i>pacifica.policy.status.transaction.files.FileLookup</i> static method), 17
<code>__init__()</code> (<i>pacifica.policy.root.Root</i> method), 20	<code>_get_instruments_for_keywords()</code> (<i>pacifica.policy.status.instrument.search.InstrumentKeywordSearch</i> method), 16
<code>__init__()</code> (<i>pacifica.policy.status.instrument_query.InstrumentQuery</i> method), 15	<code>_get_instruments_for_project()</code> (<i>pacifica.policy.status.instrument.by_project_id.InstrumentsByProject</i> static method), 16
<code>__init__()</code> (<i>pacifica.policy.status.project_query.ProjectQuery</i> method), 15	<code>_get_projects_details()</code> (<i>pacifica.policy.status.project.lookup.ProjectLookup</i> static method), 17
<code>__init__()</code> (<i>pacifica.policy.status.rest.StatusPolicy</i> method), 16	<code>_get_projects_for_keywords()</code> (<i>pacifica.policy.status.project.search.ProjectKeywordSearch</i> method), 17
<code>__init__()</code> (<i>pacifica.policy.status.transaction_query.TransactionQuery</i> method), 16	<code>_get_projects_for_user()</code> (<i>pacifica.policy.status.project.by_user.ProjectUserSearch</i> static method), 17
<code>__init__()</code> (<i>pacifica.policy.status.user_query.UserQuery</i> method), 16	<code>_get_transaction_details()</code> (<i>pacifica.policy.status.transaction.lookup.TransactionLookup</i> static method), 17
<code>_all_instrument_info()</code> (<i>pacifica.policy.admin.AdminPolicy</i> method), 19	<code>_get_transaction_list_details()</code> (<i>pacifica.policy.reporting.transaction.transaction_details.TransactionDetails</i> static method), 14
<code>_all_project_info()</code> (<i>pacifica.policy.admin.AdminPolicy</i> method), 19	<code>_get_transaction_list_summary()</code> (<i>pacifica.policy.reporting.transaction.transaction_summary.TransactionSummary</i> static method), 15
<code>_cleanup_instrument_list()</code> (<i>pacifica.policy.status.instrument.search.InstrumentKeywordSearch</i> method), 16	<code>_get_transactions_for_keywords()</code> (<i>pacifica.policy.status.transaction.search.TransactionSearch</i> static method), 18
<code>_cleanup_user_query_id()</code> (<i>pacifica.policy.uploader.rest.UploaderPolicy</i> static method), 18	<code>_get_user_info()</code> (<i>pacifica.policy.status.user.lookup.UserLookup</i> static method), 18
<code>_cleanup_object_stats()</code> (<i>pacifica.policy.reporting.transaction.transaction_summary.TransactionSummary</i> static method), 15	<code>_get_user_lookups()</code> (<i>pacifica.policy.reporting.transaction.query_base.QueryBase</i> static method), 14
<code>_filter_results()</code> (<i>pacifica.policy.uploader.rest.UploaderPolicy</i> static method), 18	<code>_get_users_for_keywords()</code> (<i>pacifica.policy.status.user.search.UserSearch</i> method), 15
<code>_format_url()</code> (<i>pacifica.policy.admin.AdminPolicy</i> method), 19	
<code>_get_available_projects()</code> (<i>pacifica.policy.status.base.QueryBase</i> method), 15	

<i>static method</i>), 18		<code>_query_select_project_info()</code>	(<i>pacif- fica.policy.uploader.rest.UploaderPolicy method</i>), 19
<code>_groups_for_inst()</code>	(<i>pacif- fica.policy.admin.AdminPolicy method</i>), 19	<code>_query_select_user_info()</code>	(<i>pacif- fica.policy.uploader.rest.UploaderPolicy method</i>), 19
<code>_instrument_info_from_ids()</code>	(<i>pacif- fica.policy.admin.AdminPolicy method</i>), 19	<code>_squash_output_list()</code>	(<i>pacif- fica.policy.status.instrument.search.InstrumentKeywordSearch static method</i>), 16
<code>_instruments_for_custodian()</code>	(<i>pacif- fica.policy.admin.AdminPolicy method</i>), 19	<code>_user_info_from_kwds()</code>	(<i>pacif- fica.policy.admin.AdminPolicy method</i>), 19
<code>_instruments_for_group()</code>	(<i>pacif- fica.policy.admin.AdminPolicy method</i>), 19	<code>_user_info_from_queries()</code>	(<i>pacif- fica.policy.uploader.rest.UploaderPolicy method</i>), 19
<code>_instruments_for_user()</code>	(<i>pacif- fica.policy.admin.AdminPolicy method</i>), 19	<code>_users_for_proj()</code>	(<i>pacif- fica.policy.admin.AdminPolicy method</i>), 19
<code>_instruments_for_user_proj()</code>	(<i>pacif- fica.policy.admin.AdminPolicy method</i>), 19	<code>_valid_query()</code>	(<i>pacif- fica.policy.ingest.rest.IngestPolicy method</i>), 14
<code>_is_admin()</code>	(<i>pacif- fica.policy.admin.AdminPolicy method</i>), 19	<code>_valid_query()</code>	(<i>pacif- fica.policy.uploader.rest.UploaderPolicy static method</i>), 19
<code>_merge_two_dicts()</code>	(<i>pacif- fica.policy.reporting.transaction.query_base.QueryBase static method</i>), 14		
<code>_object_id_valid()</code>	(<i>pacif- fica.policy.admin.AdminPolicy static method</i>), 19	A	
<code>_project_info_from_ids()</code>	(<i>pacif- fica.policy.admin.AdminPolicy method</i>), 19	AdminPolicy (<i>class in pacifica.policy.admin</i>), 19	
<code>_projects_for_custodian()</code>	(<i>pacif- fica.policy.admin.AdminPolicy method</i>), 19	<code>all_instruments_url</code>	(<i>pacif- fica.policy.admin.AdminPolicy attribute</i>), 19
<code>_projects_for_inst()</code>	(<i>pacif- fica.policy.admin.AdminPolicy method</i>), 19	<code>all_instruments_url</code>	(<i>pacif- fica.policy.status.base.QueryBase attribute</i>), 15
<code>_projects_for_user()</code>	(<i>pacif- fica.policy.admin.AdminPolicy method</i>), 19	<code>all_projects_url</code>	(<i>pacif- fica.policy.admin.AdminPolicy attribute</i>), 19
<code>_projects_for_user_inst()</code>	(<i>pacif- fica.policy.admin.AdminPolicy method</i>), 19	<code>all_projects_url</code>	(<i>pacif- fica.policy.status.base.QueryBase attribute</i>), 15
<code>_pull_data_by_rec()</code>	(<i>pacif- fica.policy.ingest.rest.IngestPolicy method</i>), 14	<code>all_relationships_url</code>	(<i>pacif- fica.policy.admin.AdminPolicy attribute</i>), 19
<code>_query_select()</code>	(<i>pacif- fica.policy.uploader.rest.UploaderPolicy method</i>), 18	<code>all_transactions_url</code>	(<i>pacif- fica.policy.status.base.QueryBase attribute</i>), 15
<code>_query_select_admin()</code>	(<i>pacif- fica.policy.uploader.rest.UploaderPolicy method</i>), 18	<code>all_users_url</code>	(<i>pacif- fica.policy.admin.AdminPolicy attribute</i>), 20
<code>_query_select_instrument_info()</code>	(<i>pacif- fica.policy.uploader.rest.UploaderPolicy method</i>), 19	B	
		<code>base_user_info</code>	(<i>pacif- fica.policy.reporting.transaction.query_base.QueryBase attribute</i>), 14

C

`create_worker_threads()` (in module `pacifica.policy.search_sync`), 21

E

`error_page_default()` (in module `pacifica.policy.root`), 20

`es_client()` (in module `pacifica.policy.search_sync`), 21

`EventsPolicy` (class in `pacifica.policy.events.rest`), 13

`exposed` (`pacifica.policy.events.rest.EventsPolicy` attribute), 13

`exposed` (`pacifica.policy.reporting.rest.ReportingPolicy` attribute), 15

`exposed` (`pacifica.policy.reporting.transaction.transaction_details.TransactionDetails` attribute), 14

`exposed` (`pacifica.policy.reporting.transaction.transaction_summary.TransactionSummary` attribute), 15

`exposed` (`pacifica.policy.root.Root` attribute), 20

`exposed` (`pacifica.policy.status.instrument.by_project_id.InstrumentsByProject` attribute), 16

`exposed` (`pacifica.policy.status.instrument.search.InstrumentKeywordSearch` attribute), 16

`exposed` (`pacifica.policy.status.instrument_query.InstrumentQuery` attribute), 15

`exposed` (`pacifica.policy.status.project.by_user.ProjectUserSearch` attribute), 17

`exposed` (`pacifica.policy.status.project.lookup.ProjectLookup` attribute), 17

`exposed` (`pacifica.policy.status.project.search.ProjectKeywordSearch` attribute), 17

`exposed` (`pacifica.policy.status.project_query.ProjectQuery` attribute), 15

`exposed` (`pacifica.policy.status.rest.StatusPolicy` attribute), 16

`exposed` (`pacifica.policy.status.transaction.files.FileLookup` attribute), 17

`exposed` (`pacifica.policy.status.transaction.lookup.TransactionLookup` attribute), 17

`exposed` (`pacifica.policy.status.transaction.search.TransactionSearch` attribute), 18

`exposed` (`pacifica.policy.status.transaction_query.TransactionQuery` attribute), 16

`exposed` (`pacifica.policy.status.user.lookup.UserLookup` attribute), 18

`exposed` (`pacifica.policy.status.user.search.UserSearch` attribute), 18

`exposed` (`pacifica.policy.status.user_query.UserQuery` attribute), 16

`exposed` (`pacifica.policy.uploader.rest.UploaderPolicy` attribute), 19

F

`FileLookup` (class in `pacifica.policy.status.transaction.files`), 17

G

`generate()` (`pacifica.policy.search_render.SearchRender` class method), 21

`generate_work()` (in module `pacifica.policy.search_sync`), 21

`GET()` (`pacifica.policy.root.Root` static method), 20

`GET()` (`pacifica.policy.status.instrument.by_project_id.InstrumentsByProject` static method), 16

`GET()` (`pacifica.policy.status.instrument.search.InstrumentKeywordSearch` method), 16

`GET()` (`pacifica.policy.status.project.by_user.ProjectUserSearch` static method), 17

`GET()` (`pacifica.policy.status.project.lookup.ProjectLookup` static method), 17

`GET()` (`pacifica.policy.status.project.search.ProjectKeywordSearch` method), 17

`GET()` (`pacifica.policy.status.transaction.files.FileLookup` static method), 17

`GET()` (`pacifica.policy.status.transaction.lookup.TransactionLookup` static method), 17

`GET()` (`pacifica.policy.status.transaction.search.TransactionSearch` static method), 18

`GET()` (`pacifica.policy.status.user.lookup.UserLookup` static method), 18

`GET()` (`pacifica.policy.status.user.search.UserSearch` method), 18

`get_config` (in module `pacifica.policy.config`), 20

`get_render_class()` (`pacifica.policy.reporting.transaction.query_base.QueryBase` static method), 14

`get_relationship_info()` (`pacifica.policy.admin.AdminPolicy` method), 20

`get_render_class()` (`pacifica.policy.search_render.SearchRender` static method), 21

`get_user_info()` (`pacifica.policy.reporting.transaction.query_base.QueryBase` static method), 14

`get_relationship_info()` (`pacifica.policy.admin.AdminPolicy` method), 20

`get_render_class()` (`pacifica.policy.search_render.SearchRender` static method), 21

`get_user_info()` (`pacifica.policy.reporting.transaction.query_base.QueryBase` static method), 14

`get_relationship_info()` (`pacifica.policy.admin.AdminPolicy` method), 20

`get_render_class()` (`pacifica.policy.search_render.SearchRender` static method), 21

`get_user_info()` (`pacifica.policy.reporting.transaction.query_base.QueryBase` static method), 14

`get_relationship_info()` (`pacifica.policy.admin.AdminPolicy` method), 20

`get_render_class()` (`pacifica.policy.search_render.SearchRender` static method), 21

M

`md_url` (*pacifica.policy.admin.AdminPolicy* attribute), 20

`md_url` (*pacifica.policy.status.base.QueryBase* attribute), 15

O

`object_exclude()` (*pacifica.policy.search_render.SearchRender* class method), 21

P

`pacifica.policy` (module), 22

`pacifica.policy.admin` (module), 19

`pacifica.policy.config` (module), 20

`pacifica.policy.events` (module), 13

`pacifica.policy.events.rest` (module), 13

`pacifica.policy.globals` (module), 20

`pacifica.policy.ingest` (module), 13

`pacifica.policy.ingest.rest` (module), 13

`pacifica.policy.reporting` (module), 14

`pacifica.policy.reporting.rest` (module), 15

`pacifica.policy.reporting.transaction` (module), 14

`pacifica.policy.reporting.transaction.query_base` (module), 14

`pacifica.policy.reporting.transaction.transaction_details` (module), 14

`pacifica.policy.reporting.transaction.transaction_summary` (module), 15

`pacifica.policy.root` (module), 20

`pacifica.policy.search_render` (module), 21

`pacifica.policy.search_sync` (module), 21

`pacifica.policy.status` (module), 15

`pacifica.policy.status.base` (module), 15

`pacifica.policy.status.instrument` (module), 16

`pacifica.policy.status.instrument.by_project_id` (module), 16

`pacifica.policy.status.instrument.search` (module), 16

`pacifica.policy.status.instrument_query` (module), 15

`pacifica.policy.status.project` (module), 17

`pacifica.policy.status.project.by_user` (module), 17

`pacifica.policy.status.project.lookup` (module), 17

`pacifica.policy.status.project.search` (module), 17

`pacifica.policy.status.project_query` (module), 15

`pacifica.policy.status.rest` (module), 16

`pacifica.policy.status.transaction` (module), 17

`pacifica.policy.status.transaction.files` (module), 17

`pacifica.policy.status.transaction.lookup` (module), 17

`pacifica.policy.status.transaction.search` (module), 18

`pacifica.policy.status.transaction_query` (module), 16

`pacifica.policy.status.user` (module), 18

`pacifica.policy.status.user.lookup` (module), 18

`pacifica.policy.status.user.search` (module), 18

`pacifica.policy.status.user_query` (module), 16

`pacifica.policy.uploader` (module), 18

`pacifica.policy.uploader.rest` (module), 18

`pacifica.policy.validation` (module), 21

`pacifica.policy.wsgi` (module), 22

`POST()` (*pacifica.policy.events.rest.EventsPolicy* method), 13

`POST()` (*pacifica.policy.ingest.rest.IngestPolicy* method), 14

`POST()` (*pacifica.policy.reporting.transaction.transaction_details.TransactionDetails* static method), 14

`POST()` (*pacifica.policy.reporting.transaction.transaction_summary.TransactionSummary* static method), 15

`POST()` (*pacifica.policy.uploader.rest.UploaderPolicy* method), 18

`proj_instrument_url` (*pacifica.policy.admin.AdminPolicy* attribute), 20

`proj_user_url` (*pacifica.policy.admin.AdminPolicy* attribute), 20

`ProjectKeywordSearch` (class in *pacifica.policy.status.project.search*), 17

`ProjectLookup` (class in *pacifica.policy.status.project.lookup*), 17

`ProjectQuery` (class in *pacifica.policy.status.project_query*), 15

`ProjectUserSearch` (class in *pacifica.policy.status.project.by_user*), 17

Q

`QueryBase` (class in *pacifica.policy.reporting.transaction.query_base*), 14

`QueryBase` (class in *pacifica.policy.status.base*), 15

R

`ReportingPolicy` (class in *pacifica*)

fica.policy.reporting.rest), 15
 Root (class in *pacifica.policy.root*), 20

S

search_sync() (in module *pacifica.policy.search_sync*), 21
 SearchRender (class in *pacifica.policy.search_render*), 21
start_work() (in module *pacifica.policy.search_sync*), 21
 StatusPolicy (class in *pacifica.policy.status.rest*), 16

T

TransactionDetails (class in *pacifica.policy.reporting.transaction.transaction_details*), 14
 TransactionLookup (class in *pacifica.policy.status.transaction.lookup*), 17
 TransactionQuery (class in *pacifica.policy.status.transaction_query*), 16
 TransactionSearch (class in *pacifica.policy.status.transaction.search*), 18
 TransactionSummary (class in *pacifica.policy.reporting.transaction.transaction_summary*), 15
try_doing_work() (in module *pacifica.policy.search_sync*), 21
try_es_connect() (in module *pacifica.policy.search_sync*), 21
try_meta_connect() (*pacifica.policy.root.Root* class method), 20

U

UploaderPolicy (class in *pacifica.policy.uploader.rest*), 18
 UserLookup (class in *pacifica.policy.status.user.lookup*), 18
 UserQuery (class in *pacifica.policy.status.user_query*), 16
 UserSearch (class in *pacifica.policy.status.user.search*), 18

V

validate_project() (in module *pacifica.policy.validation*), 21
validate_transaction() (in module *pacifica.policy.validation*), 22
validate_universal() (in module *pacifica.policy.validation*), 22
validate_user() (in module *pacifica.policy.validation*), 22

Y

yield_data() (in module *pacifica.policy.search_sync*), 21