CKAN Service Provider Documentation

Release 0.1

Open Knowledge Foundation

March 16, 2016

Contents

A simple flask app that makes functions available as synchronous or asynchronous jobs.

CHAPTER 1

Routes

GET /status
Show version, available job types and name of service.

Results:

Rtype A dictionary with the following keys

Parameters

- **version** (*float*) – Version of the service provider
- **job_types** (*list of strings*) – Available job types
- **name** (*string*) – Name of the service
- **stats** (*dictionary*) – Shows stats for jobs in queue

GET /logout
Log out the active user

POST /login
Log in as administrator

You can use wither basic auth or form based login (via POST).

Parameters

- **username** (*string*) – The administrator's username
- **password** (*string*) – The administrator's password

GET /login
Log in as administrator

You can use wither basic auth or form based login (via POST).

Parameters

- **username** (*string*) – The administrator's username
- **password** (*string*) – The administrator's password

GET /user
Show information about the current user

Rtype A dictionary with the following keys

Parameters

- **id** (*int*) – User id

- **name** (*string*) – User name
- **is_active** (*bool*) – Whether the user is currently active
- **is_anonymous** (*bool*) – The anonymous user is the default user if you are not logged in

GET /job
List all jobs.

Parameters

- **_limit** (*int*) – maximum number of jobs to show (default 100)
- **_offset** (*int*) – how many jobs to skip before showin the first one (default 0)
- **_status** (*string*) – filter jobs by status (complete, error)

Also, you can filter the jobs by their metadata. Use the metadata key as parameter key and the value as value.

Rtype A list of job ids

DELETE /job
Clear old jobs

Parameters

- **days** (*integer*) – Jobs for how many days should be kept (default: 10)

Status Codes

- 200 OK – no error
- 403 Forbidden – not authorized to delete jobs
- 409 Conflict – an error occurred

POST /job/ (*job_id*)

POST /job
Submit a job. If no id is provided, a random id will be generated.

Parameters

- **job_type** (*string*) – Which kind of job should be run. Has to be one of the available job types.
- **api_key** (*string*) – An API key that is needed to execute the job. This could be a CKAN API key that is needed to write any data. The key will also be used to administer jobs. If you don't want to use a real API key, you can provide a random string that you keep secure.
- **data** (*json encodable data*) – Data that is send to the job as input. (Optional)
- **result_url** (*url string*) – Callback url that is called once the job has finished. (Optional)
- **metadata** (*list of key - value pairs*) – Data needed for the execution of the job which is not the input data. (Optional)

Results:

Rtype A dictionary with the following keys

Parameters

- **job_id** (*string*) – An identifier for the job
- **job_key** (*string*) – A key that is required to view and administer the job

Status Codes

- 200 OK – no error
- 409 Conflict – an error occurred

GET /
Show link to documentation.

Rtype A dictionary with the following keys

Parameters

- **help** (*string*) – Help text

GET /job/ (*job_id*) **/data**
Get the raw data that the job returned. The mimetype will be the value provided in the metdata for the key `mimetype`.

Results:

Rtype string

Status Codes

- 200 OK – no error
- 403 Forbidden – not authorized to view the job's data
- 404 Not Found – job id not found
- 409 Conflict – an error occurred

GET /job/ (*job_id*)
Show a specific job.

Results:

Rtype A dictionary with the following keys

Parameters

- **status** (*string*) – Status of job (complete, error)
- **sent_data** (*json encodable data*) – Input data for job
- **job_id** (*string*) – An identifier for the job
- **result_url** (*url string*) – Callback url
- **data** (*json encodable data*) – Results from job.
- **error** (*string*) – Error raised during job execution
- **metadata** (*list of key - value pairs*) – Metadata provided when submitting job.
- **requested_timestamp** (*timestamp*) – Time the job started
- **finished_timestamp** (*timestamp*) – Time the job finished

Status Codes

- 200 OK – no error
- 403 Forbidden – not authorized to view the job's data
- 404 Not Found – job id not found
- 409 Conflict – an error occurred

DELETE /job/(*job_id*)

Deletes the job together with its logs and metadata.

Parameters

- **job_id** (*string*) – An identifier for the job

Status Codes

- 200 OK – no error
- 403 Forbidden – not authorized to delete the job
- 404 Not Found – the job could not be found
- 409 Conflict – an error occurred

CHAPTER 2

Administration

To view the results of a job or resubmit it, the job key, that is returned when a job is created, is needed. Alternatively, you can log in as admin or provide the secure key. The credentials for the admin user and the secure key stored in the settings file.

CHAPTER 3

Add a job

Just decorate your function and it will become available as a job:

```
import ckanserviceprovider.job as job
import ckanserviceprovider.util as util

@job.sync
def echo(task_id, input):
    handler = util.StoringHandler(task_id, input)
    logger = logging.getLogger(__name__)
    logger.addHandler(handler)

    if input['data'].startswith('>'):
        raise util.JobError('do not start message with >')
    if input['data'].startswith('#'):
        raise Exception('serious exception')
    if input['data'].startswith('&'):
      logger.warn('just a warning')
    return '>' + input['data']
```

Expected job errors should be raised as *util.JobError*. For logging, use the handler `util.StoringHandler` to make sure that the logs are properly saved.

/

GET /, 5

/job

GET /job, 4
GET /job/(job_id), 5
GET /job/(job_id)/data, 5
POST /job, 4
POST /job/(job_id), 4
DELETE /job, 4
DELETE /job/(job_id), 5

/login

GET /login, 3
POST /login, 3

/logout

GET /logout, 3

/status

GET /status, 3

/user

GET /user, 3