ClaimStore Documentation

Release 0.1.0

Invenio Collaboration

January 27, 2017

Contents

CHAPTER 1

About

This document describes ClaimStore, a new proposed mini service that permits to exchange information about information claims within a set of collaborating heterogeneous information services.

A primary use case example is the exchange of information about persistent identifiers (such as DOI) in the domain of astrophysics and particle physics among ADS, arXiv, and INSPIRE services. This would permit to better track and disambiguate papers, offer cross-linking between services, or offer cross-search between services.

The driving idea behind ClaimStore is to offer a neutral micro-service that would be (1) storing claims that various collaborating services perform and (2) answering questions about claims.

CHAPTER 2

Table of contents

2.1 Installation

2.1.1 Using Docker

```
$ docker-compose build
$ docker-compose run --rm web bower install
$ docker-compose run --rm web claimstore database create
$ docker-compose run --rm web claimstore database populate  # optional
$ docker-compose run --rm web /code/run-tests.sh
$ docker-compose up
```

2.1.2 Using command line

```
$ mkvirtualenv claimstore --python=$(which python3.4)
$ sudo apt-get install npm  # install nodejs
$ sudo npm install -g bower
$ pip install -e .[tests,docs]
$ bower install
$ export SQLALCHEMY_DATABASE_URI=postgres://postgres:postgres@db:5432/postgres
$ claimstore database create
$ claimstore database populate  # optional
$ python setup.py test
$ claimstore run
```

2.2 Description

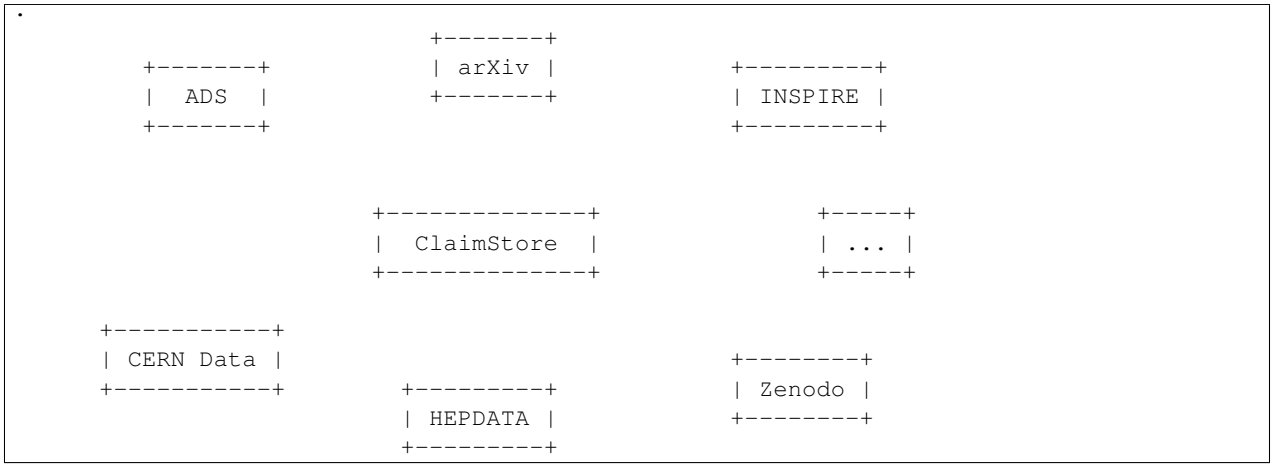
This part of the documentation presents technical description of the ClaimStore system.

> *Consider a collaborative system of independent digital library heterogeneous services (S1, S2, ...) that want to exchange information about the data objects (O1, O2, ...) of various types (T1, T2, ...) that they manage.*

For example, the collaborative network of ADS, arXiv, and INSPIRE that exchanges information about papers and people using arXiv IDs, ADS bibcodes, ORCID, DOI persistent identifier types.

2.2.1 Defining collaborative network

The collaborative network of services is defined means of describing each service and each persistent object types that they usually manage. Each participating service in the network, such as:



will describe the assets it manages, for example, INSPIRE manages record IDs and person IDs:

```
{
  "service": "INSPIRE",
  "url": "http://inspirehep.net",
  "persistent_identifiers": [
     {
       "type": "INSPIRE_RECORD_ID",
       "description": "INSPIRE record",
       "url": "http://inspirehep.net/record/<INSPIRE_RECORD_ID>",
       "example_value": "123",
       "example_url": "http://inspirehep.net/record/123",
     },
     {
       "type": "INSPIRE_AUTHOR_ID",
       "description": "INSPIRE author",
       "url": "http://inspirehep.net/author/<INSPIRE_AUTHOR_ID>",
       "example_value": "J.R.Ellis.1",
       "example_url": "http://inspirehep.net/author/J.R.Ellis.1",
     },
  ],
}
```

Optionally, the service can also expose some information about how the claims are asserted, for example:

```
{
  "service": "INSPIRE",
  "url": "http://inspirehep.net",
  "persistent_identifiers": [
     {
       "type": "INSPIRE_RECORD_ID",
       "description": "INSPIRE record",
       "url": "http://inspirehep.net/record/<INSPIRE_RECORD_ID>",
       "example_value": "123",
       "example_url": "http://inspirehep.net/record/123",
     },
     {
       "type": "INSPIRE_AUTHOR_ID",
```

```
        "description": "INSPIRE author",
        "url": "http://inspirehep.net/author/<INSPIRE_AUTHOR_ID>",
        "example_value": "J.R.Ellis.1",
        "example_url": "http://inspirehep.net/author/J.R.Ellis.1",
      },
  ],
  "certainty_levels": [
    {
      "1.0": "human approved"
    },
    {
      "0.5": "trusted algorithm"
    },
    {
      "0.1": "less trusted algorithm"
    },
    {
      "0.0": "guess"
    }
  ]
}
```

Defining all services in this way will define our (1) operational service nework (ADS, arXiv, INSPIRE, HEPDATA, etc), (2) data objects (papers, people, software, data, etc) and (3) persistent identifier types (arXIv ID, ADS bibcode, ORCID, DOI, etc) that the network uses.

2.2.2 Making claims

Each service can make claims about objects, for example:

> *Service S1 says that object O1 of type T1 is the same as the object O2 of type T2 with a certainty of C.*

For example, ADS can claim that `arXiv:astro-ph/0501001` is having bibcode `2005astro.ph..1001H`:

```
{
  "claimant": "ADS",
  "subject": {
      "type": "ARXIV_ID",
      "value": "astro-ph/0501001"
   },
  "predicate": "is_same_as",
  "certainty": 1,
  "object": {
      "type": "ADS_BIBCODE",
      "value": "2005astro.ph..1001H"
  },
  "created": "2015-05-26T11:00:00Z"
}
```

Each individual claim can optionally include a free set of additional parameters detailing the claim, for example:

> *... as was asserted on day D1 using algorithm A1 with parameters P1, P2, P3 and subsequently verified by humans H1 and H2 using external databases E1 and E2.*

For example, we can say that the ADS bibcode was added automatically by a trusted program:

```
{
  "claimant": "ADS",
  "subject": {
```

```
      "type": "ARXIV_ID",
      "value": "astro-ph/0501001"
   },
  "predicate": "is_same_as",
  "certainty": 0.9,
  "object": {
      "type": "ADS_BIBCODE",
      "value": "2005astro.ph..1001H"
  },
  "arguments": {
      "human": 0,
      "actor": "ADS_record_generator"
  },
  "created": "2015-05-26T11:00:00Z"
}
```

A service would usually claim something about the objects it manages. In the following example, CDS claims that "CMS-PAS-HIG-14-008" has a persistent CDS record ID 2001192:

```
{
  "claimant": "CDS",
  "subject": {
      "type": "CDS_REPORT_NUMBER",
      "value": "CMS-PAS-HIG-14-008"
   },
  "predicate": "is_same_as",
  "certainty": 1,
  "object": {
      "type": "CDS_RECORD_ID",
      "value": "2001192"
  },
  "arguments": {
      "human": 0,
      "actor": "CDS_submission"
  },
  "created": "2015-05-26T11:00:00Z"
}
```

A service can claim statements about holdings of other services in the the collaborative network. For example, INSPIRE can claim that the arXiv paper "cond-mat/9906097" is having DOI of "10.1103/PhysRevE.62.7422" with high certainty, as it was confirmed by an apprentice cataloguer:

```
{
  "claimant": "INSPIRE",
  "subject": {
      "type": "ARXIV_ID",
      "value": "cond-mat/9906097"
   },
  "predicate": "is_same_as",
  "certainty": 0.8,
  "object": {
      "type": "DOI",
      "value": "10.1103/PhysRevE.62.7422"
  },
  "arguments": {
      "human": 1,
      "actor": "John Doe",
      "role": "cataloguer"
  },
```

```
  "created": "2015-05-26T11:00:00Z"
}
```

2.2.3 Using claims

Each participating service can ask questions about claims related to individual objects, such as:

> *Who knows anything about DOI "10.1103/PhysRevE.62.7422"?*

which would be asked via:

```
GET /claims/?type=DOI&value=10.1103/PhysRevE.62.7422
```

Upon seeing this query, the ClaimStore would return a list of claims about this DOI (whether found as a subject or an object of the claim), in chronological order, for example:

```
[
  {
    "claimant": "INSPIRE",
    "subject": {
        "type": "ARXIV_ID",
        "value": "cond-mat/9906097"
     },
    "predicate": "is_same_as",
    "certainty": 0.8,
    "object": {
        "type": "DOI",
        "value": "10.1103/PhysRevE.62.7422"
    },
    "arguments": {
        "human": 1,
        "actor": "John Doe",
        "role": "cataloguer"
    },
    "created": "2015-05-26T11:00:00Z",
    "recieved": "2015-05-26T11:00:00Z"
  },
  {
    "claimant": "ARXIV",
    "subject": {
        "type": "ARXIV_ID",
        "value": "cond-mat/9906097"
     },
    "predicate": "is_same_as",
    "certainty": 1.0,
    "object": {
        "type": "DOI",
        "value": "10.1103/PhysRevE.62.7422"
    },
    "arguments": {
        "human": 1,
        "actor": "John Doe",
        "role": "author"
    },
    "created": "2015-05-26T11:00:00Z",
    "recieved": "2015-05-26T11:00:00Z"
  },
]
```

ClaimStore will faithfully return the list of any claims it knows about this DOI without manipulating them.

Each service can ask summary questions as well, such as:

> *What did CERN Open Data ever said about software packages with high confidence?*

which would be asked via:

```
GET /claims/?claimant=CERNOPENDATA&type=SOFTWARE&confidence=50+
```

More complex querying on the JSON structure of claims can be done, for example:

```
*Which claims were done by John Doe from INSPIRE between 2012-01-23
and 2012-08-07?*
```

which would be asked via:

```
GET /claims/?claimant=INSPIRE&since=2012-01-03&until=2012-08-07&claim.arguments.actor=John%20Doe
```

e.g. because we learned that the procedure was buggy at the time and would like to clean it.

Any such possible evolution depends on the further uses of the system beyond simple persistent ID exchange.

2.2.4 Managing claims

ClaimStore is a neutral application dedicated to efficiently storing individual claims and answering questions about them. ClaimStore *does not* attempt to impose any workflow or resolve any possible conflicts, such as when service S1 claims that object O1 is the same as object O2 with certainty C1, while service S2 claims that object O1 is the same as object O3 with certainty C2. The resolution of conflicts and is left upon each participating service that can implement a solution fitting its own workflows and quality standards.

For example, when INSPIRE receives an arXiv paper of the "astro-ph" category, it can ask ClaimStore about all the claims related to it:

```
GET /claims/?type=ARXIV_ID&value=arXiv:1505.06718&claimant=ADS
```

as it may decide to trust ADS's claims more than author claims or publisher claims in this subject domain.

If a service wants to revoke an old claim, it can make a new claim with higher certainty.

The bottom line is that ClaimStore does not attempt to do any judgement about claims, nor does it do any management of claims beyond simply storing what the services claimed and answering questions about stored assets.

2.2.5 Notifications

The usual usage of ClaimStore by the services is (1) pushing own claim information to the ClaimStore in order to register new claims and (2) pulling information about others' claims from the ClaimStore as the service needs them.

Alternatively, another mode of service operation could include (3) registering to be automatically notified via push notifications in case somebody claims something about a certain object types. This could come as a later extension.

2.2.6 Authorisations

After a service registers in the collaborative network, it is given a secret key that the service could use to push the claim information.

Each participating service is allowed to read claims made by others.

This would be sufficient for a simple start of the service. A possible extension could include (1) opening parts of the claim database for other non-participating clients, or (2) introducing trusted partners making claims on others' behalf, etc.

2.3 Design

This part of the documentation presents general architecture, design and implementation guidelines.

2.3.1 Architecture

ClaimStore is an independent mini-application built upon our usual Flask ecosystem:

- Flask-RESTful for REST API
- Flask-Notifications for optional alerts
- OAuth for authorisation needs
- SQLAlchemy for DB abstraction
- JSON Schema for JSON object description
- PostgreSQL for DB persistence and JSON search

2.3.2 Database

The information about network of services, data objects and persistent identifier types, and claims about them is described via JSON snippets.

The JSON data is stored in several tables for `claimants`, `object_types` etc. The individual claims are stored in a `claims` table that uses both regular RDBMS and JSONB columns, permitting some fast inter-table JOINs as well as free-format additional claim parameters, for example:

```
claims
======================
uuid            integer
created         date
claimant        ref ->
subject_type    ref ->
subject_value   text
claim           ref ->
certainty       number
claim_details   jsonb
status          ref ->  e.g. to mark revoked claims
object_type     ref ->
object_value    text
```

The JSON format of claims is also checked against a formal JSON schema to verify its validity upon claim submission. There are several JSON Schemata describing the system: one JSON schema describes a service, another JSON schema describes a persistent ID type, another JSON schema describes a claim, etc.

For searching the claim database, PostgreSQL/JSONB column type can be used which offers efficient querying out of the box. In case of extended usage needs, JSON claims can be propagated to an Elasticsearch cluster. that can increase query speed and query language further.

2.3.3 Claim types

The primary motivation behind ClaimStore was the exchange of information about persistent identifiers, hence the typical claim types are:

- *is_same_as*: used when there is a 100% equivalence, e.g. a local copy of an arXiv record, with either the same or enriched metadata, e.g. ORCID corresponds to this INSPIRE ID
- *is_variant_of*: lesser claim, e.g. arXiv preprint and DOI of a published paper, e.g. when INSPIRE merges two sources into one

However, the system is generic enough to accept any kind of claims, so the ClaimStore can also be used to store information about other types of relations, such as:

- *is_author_of*: this person is the author of this document
- *is_contributor_to*: this person is supervisor/translator/spokesperson of this document
- *is_erratum_of*: e.g. if INSPIRE record R1 is variant of DOI1, and DOI2 is erratum of DOI1, but INSPIRE merges all these in the same record, then there would be three claims: R1 is variant of DOI1, DOI2 is erratum of DOI1, R1 is variant of DOI2

Examples of other possible relations that could be included in the future are:

- `is_cited_by`
- `is_superseded_by`
- `is_software_for_paper`
- `is_dataset_for_paper`
- `is_dataset_for_software`

For example, imagine the following table of claims:

```
subject               predicate      object
--------------------  -----------    -------------------
arXiv:hep-th/0101001  is_variant_of  DOI:10.1234/foo.bar
arXiv:hep-th/0101001  is_same_as     arXiv:1506.07188
```

One could then ask queries like *who does know about DOI 10.1234/foo.bar?* and the system could return only direct claims:

```
GET /claims/?type=DOI&value=10.1234/foo.bar
```

listing only the first relation, or else we could also ask to include all indirect claims:

```
GET /claims/?type=DOI&value=10.1234/foo.bar&include=indirect&certainty=0.5+
```

which would return both relations.

2.4 Releases

This part of the documentation presents history of ClaimStore releases.

2.4.1 Changes

Version 0.1.0 (released TBD)

- Initial public release.

2.5 Restful Resources

2.5.1 Submit a claimant

Register a new claimant in the ClaimStore.

POST /api/claimants

This resource is expecting JSON data with all the necessary information of a new claimant.

Request:

```
POST /api/claimants HTTP/1.1
Content-Type: application/json
Host: localhost:5000

{
    "name": "INSPIRE",
    "url": "http://inspirehep.net"
}
```

Request Headers

- Content-Type – application/json

JSON Parameters

- **body** – JSON with the information of the claimant. The JSON data should be valid according to the JSON Schema for claimants.

Responses:

```
HTTP/1.0 200 OK
Content-Length: 80
Content-Type: application/json

{
    "status": "success",
    "uuid": "ab19c98b-xxxx-xxxx-xxxx-1d6af3bf58b4"
}
```

```
HTTP/1.0 400 BAD REQUEST
Content-Length: 95
Content-Type: application/json

{
    "extra": null,
    "message": "This claimant is already registered",
    "status": 400
}
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – no error - the claimant was subscribed
- 400 Bad Request – invalid request - problably a malformed JSON

- 403 Forbidden – access denied

Usage:

- From python:

```
import json
import requests

url = "http://localhost:5000/api/claimants/
headers = {"Content-Type": "application/json"}
with open(
    '$PROJECT_HOME/tests/config/claimants/cds.json'
) as f:
    data=json.dumps(f.read())
r = requests.post(url, data=data, headers=headers)
```

- From httpie:

```
$ http POST http://localhost:5000/api/claimants < tests/myclaimstore/config/claimants/cds.js
```

- From curl:

```
$ curl http://localhost:5000/api/claimants \
       -H "Content-Type: application/json" \
       -d @tests/myclaimstore/config/claimants/inspire.json -X POST -v
```

2.5.2 List claimants

GET service that returns all the available claimants.

GET /api/claimants/ (**uuid:** *claimant_id*)
Returns a JSON list with all the claimants.

Request:

```
GET /api/claimants HTTP/1.1
Accept: */*
Host: localhost:5000
```

```
GET /api/claimants/0e64606e-68ce-482e-ad59-1e99... HTTP/1.1
Accept: */*
Host: localhost:5000
```

Request Headers

- Content-Type – application/json

Response:

```
HTTP/1.0 200 OK
Content-Length: 108
Content-Type: application/json

[
    {
        "name": "ADS",
        "url": "http://adsabs.harvard.edu/",
        "uuid": "00000000-be54-45c0-89fa-00000000"
```

```
        },
        {
            "name": "ARXIV",
            "url": "http://arxiv.org/",
            "uuid": "00000000-87a6-4095-99a9-00000000"
        },
        {
            "name": "CDS",
            "url": "http://cds.cern.ch",
            "uuid": "00000000-602e-4912-8a9b-00000000"
        },
        {
            "name": "INSPIRE",
            "url": "http://inspirehep.net",
            "uuid": "00000000-42a5-4d3f-b54a-00000000"
        }
]
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – no error
- 400 Bad Request – invalid request
- 403 Forbidden – access denied

Usage:

- From python:

```
import requests
response = requests.get("http://localhost:5000/api/claimants")
print response.json()
```

- From httpie:

```
$ http GET http://localhost:5000/api/claimants
```

- From curl:

```
$ curl http://localhost:5000/api/claimants
```

2.5.3 Submit a claim

Record a new claim.

POST /api/claims

This resource is expecting JSON data with all the necessary information of a new claim.

Request:

```
POST /api/claims HTTP/1.1
Accept: application/json
Content-Length: 336
Content-Type: application/json
```

```
{
    "arguments": {
        "actor": "CDS_submission",
        "human": 0
    },
    "certainty": 1.0,
    "claimant": "CDS",
    "created": "2015-03-25T11:00:00Z",
    "object": {
        "type": "CDS_REPORT_NUMBER",
        "value": "CMS-PAS-HIG-14-008"
    },
    "predicate": "is_same_as",
    "subject": {
        "type": "CDS_RECORD_ID",
        "value": "2003192"
    }
}
```

Request Headers

- Content-Type – application/json

JSON Parameters

- **body** – JSON with the information of the claimt. The JSON data should be valid according to the JSON Schema for claims.

Responses:

```
HTTP/1.0 200 OK
Content-Length: 80
Content-Type: application/json

{
    "status": "success",
    "uuid": "fad4ec9f-0e95-4a22-b65c-d01f15aba6be"
}
```

```
HTTP/1.0 400 BAD REQUEST
Content-Length: 9616
Content-Type: application/json
Date: Tue, 22 Sep 2015 09:02:25 GMT
Server: Werkzeug/0.10.4 Python/3.4.3

{
    "extra": "'claimant' is a required property. Failed
              validating 'required' in schema...",
    "message": "JSON data is not valid",
    "status": 400
}
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – no error - the claim was recorded

- 400 Bad Request – invalid request - problably a malformed JSON
- 403 Forbidden – access denied

Usage:

- From python:

```
import json
import requests

url = "http://localhost:5000/api/claims/
headers = {"Content-Type": "application/json"}
with open(
    '$PROJECT_HOME/tests/config/claims/cds.1.json'
) as f:
    data=json.dumps(f.read())
r = requests.post(url, data=data, headers=headers)
```

- From httpie:

```
$ http POST http://localhost:5000/api/claims < tests/myclaimstore/data/claims/cds.1.json
```

- From curl:

```
$ curl http://localhost:5000/api/claims \
       -H "Content-Type: application/json" \
       -d @tests/myclaimstore/data/claims/inspire.1.json -X POST -v
```

2.5.4 List claims

GET service that returns the stored claims.

GET /api/claims/ (**uuid:** *claim_id*)
Returns a JSON list with all the claims matching the query parameters.

Request:

```
GET /api/claims?type=CDS_RECORD_ID&value=cond-mat/9906097&
recurse=1 HTTP/1.1
Accept: */*
Host: localhost:5000
```

```
GET /api/claims/0e64606e-68ce-482e-ad59-1e9981394f HTTP/1.1
Accept: */*
Host: localhost:5000
```

Request Headers

- Content-Type – application/json

Query Parameters

- **since** (*datetime*) – it must have the format 'YYYY-MM-DD'. It fetches claims that were created from this given datetime.
- **until** (*datetime*) – it must have the format 'YYYY-MM-DD'. It fetches claims that were created up to this given datetime.
- **claimant** (*string*) – claimant's unique name. It fetches claims submitted by the specified claimant.

- **predicate** (*string*) – predicate's unique name. It finds claims using this predicate (e.g. is_same_as).
- **certainty** (*float*) – float number between 0 and 1.0. It will search for claims with at least the specified certainty.
- **human** (*int*) – enter 1 if searching for human-created claims, 0 for algorithms and nothing in order to retrieve all.
- **actor** (*string*) – it filters claims by their actor's name (one can use %).
- **role** (*string*) – it filters claims by their actor's role (one can use %).
- **type** (*string*) – it finds claims using a certain identifier type (either subject or object). For instance: DOI.
- **value** (*string*) – it fetches all the claims with that identifier value.
- **recurse** (*boolean*) – used in combination with *type* and *value* will find all the equivalent identifiers to the specified one.
- **subject** (*string*) – it fetches claims using the given identifier type as a subject type.
- **object** (*string*) – it fetches claims using the given identifier type as an object type.
- **page** (*int*) – page from which to fetch data.
- **per_page** (*int*) – amount of data per page.

Response:

```
HTTP/1.0 200 OK
Content-Length: 1166
Content-Type: application/json

[
    {
        "arguments": {
            "actor": "CDS_submission",
            "human": 0
        },
        "certainty": 1.0,
        "claimant": "CDS",
        "created": "2015-03-25T11:00:00Z",
        "object": {
            "type": "CDS_REPORT_NUMBER",
            "value": "CMS-PAS-HIG-14-008"
        },
        "predicate": "is_same_as",
        "recieved": "2015-09-22T08:18:30.606912+00:00",
        "subject": {
            "type": "CDS_RECORD_ID",
            "value": "2003192"
        },
        "uuid": "44103ee2-0d87-47f9-b0e4-77673d297cdb"
    },
    {
        "arguments": {
            "actor": "John Doe",
            "human": 1,
            "role": "cataloguer"
        },
```

```
                "certainty": 0.5,
                "claimant": "INSPIRE",
                "created": "2015-05-25T11:00:00Z",
                "object": {
                    "type": "CDS_RECORD_ID",
                    "value": "2003192"
                },
                "predicate": "is_variant_of",
                "recieved": "2015-09-22T08:18:30.638933+00:00",
                "subject": {
                    "type": "INSPIRE_RECORD_ID",
                    "value": "cond-mat/9906097"
                },
                "uuid": "27689445-02b9-4d5d-8f9b-da21970e2352"
            }
        ]
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – no error
- 400 Bad Request – invalid request
- 403 Forbidden – access denied

Usage:

- From python:

```
import requests
response = requests.get("http://localhost:5000/api/claims")
print response.json()
```

- From httpie:

```
$ http GET http://localhost:5000/api/claims
```

- From curl:

```
$ curl http://localhost:5000/api/claims
```

2.5.5 List identifiers

GET service that returns the stored identifiers.

GET /api/identifiers
Returns a JSON list with all the available identifiers.

Request:

```
GET /api/identifiers HTTP/1.1
Accept: */*
Host: localhost:5000
```

Request Headers

- Content-Type – application/json

Response:

```
HTTP/1.0 200 OK
Content-Length: 147
Content-Type: application/json

[
    "ARXIV_ID",
    "CDS_AUTHOR_ID",
    "CDS_RECORD_ID",
    "CDS_REPORT_NUMBER",
    "DOI",
    "INSPIRE_AUTHOR_ID",
    "INSPIRE_RECORD_ID"
]
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – no error
- 400 Bad Request – invalid request
- 403 Forbidden – access denied

Usage:

- From python:

```
import requests
response = requests.get("http://localhost:5000/api/identifiers")
print response.json()
```

- From httpie:

```
$ http GET http://localhost:5000/api/identifiers
```

- From curl:

```
$ curl http://localhost:5000/api/identifiers
```

2.5.6 List predicates

GET service that returns all the available predicates.

GET /api/predicates
Returns a JSON list with all the predicates.

Request:

```
GET /api/predicates HTTP/1.1
Accept: */*
Host: localhost:5000
```

Request Headers

- Content-Type – application/json

Response:

```
HTTP/1.0 200 OK
Content-Length: 108
Content-Type: application/json

[
    "is_author_of",
    "is_contributor_to",
    "is_erratum_of",
    "is_same_as",
    "is_variant_of"
]
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – no error
- 400 Bad Request – invalid request
- 403 Forbidden – access denied

Usage:

- From python:

```
import requests
response = requests.get("http://localhost:5000/api/predicates")
print response.json()
```

- From httpie:

```
$ http GET http://localhost:5000/api/predicates
```

- From curl:

```
$ curl http://localhost:5000/api/predicates
```

2.5.7 List equivalent identifiers

GET service that returns all the stored Equivalent Identifiers.

GET /api/eqids/ (**uuid:** *eqid*)
Returns all the type/value entries in the index grouped by their equivalent identifiers.

Requests:

```
GET /api/eqids HTTP/1.1
Accept: */*
Host: localhost:5000
```

```
GET /api/eqids/0e64606e-68ce-482e-ad59-1e9981394f8 HTTP/1.1
Accept: */*
Host: localhost:5000
```

Request Headers

- Content-Type – application/json

Parameters

- **eqid** – query by a specific uuid which is shared by some equivalent identifiers

Response:

```
HTTP/1.0 200 OK
Content-Length: 592
Content-Type: application/json

{
    "36dfb125-5c35-4d3a-870c-76eb4bad498e": [
        {
            "type": "ARXIV_ID",
            "value": "cond-mat/9906097"
        },
        {
            "type": "DOI",
            "value": "C10.1103/PhysRevE.62.7422"
        }
    ],
    "77c4a5eb-3ed8-4c80-ba0d-644d6bc397a3": [
        {
            "type": "CDS_RECORD_ID",
            "value": "2003192"
        },
        {
            "type": "CDS_REPORT_NUMBER",
            "value": "CMS-PAS-HIG-14-008"
        },
        {
            "type": "INSPIRE_RECORD_ID",
            "value": "cond-mat/9906097"
        }
    ]
}
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – no error
- 400 Bad Request – invalid request - problably a malformed UUID
- 403 Forbidden – access denied

Usage:

- From python:

```
import requests
response = requests.get("http://localhost:5000/api/eqids")
print response.json()
```

- From httpie:

```
$ http GET http://localhost:5000/api/eqids
```

- From curl:

```
$ curl http://localhost:5000/api/eqids
```

2.6 Developers

This part of the documentation presents meta-information useful for ClaimStore developers and development process.

2.6.1 Contributing

Bug reports, feature requests, and other contributions are welcome. If you find a demonstrable problem that is caused by the code of this library, please:

1. Search for already reported problems.
2. Check if the issue has been fixed or is still reproducible on the latest *master* branch.
3. Create an issue with **a test case**.

If you create a feature branch, you can run the tests to ensure everything is operating correctly:

```
$ ./run-tests.sh
```

You can also test your feature branch using Docker:

```
$ docker-compose build
$ docker-compose run --rm web /code/run-tests.sh
```

2.6.2 Authors

Contact us at info@invenio-software.org

- Jose Benito Gonzalez Lopez <jose.benito.gonzalez@cern.ch>
- Tibor Simko <tibor.simko@cern.ch>

2.6.3 License

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
```

```
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

  To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights.  Therefore, you have
certain responsibilities if you distribute copies of the software, or if
you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received.  You must make sure that they, too, receive
or can get the source code.  And you must show them these terms so they
know their rights.

  Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License
giving you legal permission to copy, distribute and/or modify it.

  For the developers' and authors' protection, the GPL clearly explains
that there is no warranty for this free software.  For both users' and
authors' sake, the GPL requires that modified versions be marked as
changed, so that their problems will not be attributed erroneously to
authors of previous versions.

  Some devices are designed to deny users access to install or run
modified versions of the software inside them, although the manufacturer
can do so.  This is fundamentally incompatible with the aim of
protecting users' freedom to change the software.  The systematic
pattern of such abuse occurs in the area of products for individuals to
use, which is precisely where it is most unacceptable.  Therefore, we
have designed this version of the GPL to prohibit the practice for those
products.  If such problems arise substantially in other domains, we
stand ready to extend this provision to those domains in future versions
of the GPL, as needed to protect the freedom of users.

  Finally, every program is threatened constantly by software patents.
States should not allow patents to restrict development and use of
software on general-purpose computers, but in those that do, we wish to
avoid the special danger that patents applied to a free program could
make it effectively proprietary.  To prevent this, the GPL assures that
patents cannot be used to render the program non-free.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS
```

```
  0. Definitions.

  "This License" refers to version 3 of the GNU General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License.  Each licensee is addressed as "you".  "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy.  The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy.  Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.

  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it.  "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
```

```
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
permission to run the unmodified Program.  The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work.  This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
not control copyright.  Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
```

```
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under section
    7.  This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.
```

```
  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.

    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source.  This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
    with subsection 6b.

    d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to the
    Corresponding Source in the same way through the same place at no
    further charge.  You need not require recipients to copy the
    Corresponding Source along with the object code.  If the place to
    copy the object code is a network server, the Corresponding Source
    may be on a different server (operated by you or a third party)
    that supports equivalent copying facilities, provided you maintain
    clear directions next to the object code saying where to find the
    Corresponding Source.  Regardless of what server hosts the
    Corresponding Source, you remain obligated to ensure that it is
    available for as long as needed to satisfy these requirements.

    e) Convey the object code using peer-to-peer transmission, provided
    you inform other peers where the object code and Corresponding
    Source of the work are being offered to the general public at no
    charge under subsection 6d.

  A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

  A "User Product" is either (1) a "consumer product", which means any
tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation
into a dwelling.  In determining whether a product is a consumer product,
doubtful cases shall be resolved in favor of coverage.  For a particular
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the status
of the particular user or of the way in which the particular user
```

```
actually uses, or expects or is expected to use, the product.  A product
is a consumer product regardless of whether the product has substantial
commercial, industrial or non-consumer uses, unless such uses represent
the only significant mode of use of the product.

  "Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to install
and execute modified versions of a covered work in that User Product from
a modified version of its Corresponding Source.  The information must
suffice to ensure that the continued functioning of the modified object
code is in no case prevented or interfered with solely because
modification has been made.

  If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information.  But this requirement does not apply
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

  The requirement to provide Installation Information does not include a
requirement to continue to provide support service, warranty, or updates
for a work that has been modified or installed by the recipient, or for
the User Product in which it has been modified or installed.  Access to a
network may be denied when the modification itself materially and
adversely affects the operation of the network or violates the rules and
protocols for communication across the network.

  Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.

  7. Additional Terms.

  "Additional permissions" are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law.  If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

  When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it.  (Additional permissions may be written to require their own
removal in certain cases when you modify the work.)  You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders of
```

```
that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or

    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or

    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.

  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.
```

```
  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

  Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License.  If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or
run a copy of the Program.  Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance.  However,
nothing other than this License grants you permission to propagate or
modify any covered work.  These actions infringe copyright if you do
not accept this License.  Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.

  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License.  You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations.  If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License.  For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based.  The
work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
```

```
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version.  For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement).  To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.

  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients.  "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License.  You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
```

```
otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all.  For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

  13. Use with the GNU Affero General Public License.

  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a single
combined work, and to convey the resulting work.  The terms of this
License will continue to apply to the part which is the covered work,
but the special requirements of the GNU Affero General Public License,
section 13, concerning interaction through a network will apply to the
combination as such.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

  Each version is given a distinguishing version number.  If the
Program specifies that a certain numbered version of the GNU General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation.  If the Program does not specify a version number of the
GNU General Public License, you may choose any version ever published
by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

  Later license versions may give you additional or different
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
```

```
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:
```

```
    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<http://www.gnu.org/philosophy/why-not-lgpl.html>.
```

2.7 claimstore

2.7.1 claimstore package

Subpackages

claimstore.core package

Subpackages

claimstore.core.db package

Submodules

claimstore.core.db.types module Custom database types.

class claimstore.core.db.types.**UTCDateTime**(**args*, ***kwargs*)

Bases: sqlalchemy.sql.type_api.TypeDecorator

Custom UTC DateTime type.

impl

alias of DateTime

Process binding.

Process result.

Module contents Database related modules.

Submodules

claimstore.core.datetime module DateTime related tools.

claimstore.core.datetime.**loc_date_utc**(*date*)
Localise a naive date in UTC.

Parameters **date** (*datetime.*) – naive date

Returns date with UTC tzinfo

Return type datetime.

claimstore.core.datetime.**now_utc**()
Return UTC datetime with tzinfo.

claimstore.core.exception module Definition of specific exceptions.

exception claimstore.core.exception.**InvalidJSONData**(*message*, *status_code=None*, *extra=None*)
Bases: *claimstore.core.exception.RestApiException*

Invalid JSON Data.

This exception is raised when there is some JSON data that does not follow its associated JSON schema.

exception claimstore.core.exception.**InvalidRequest**(*message*, *status_code=None*, *extra=None*)
Bases: *claimstore.core.exception.RestApiException*

REST request could not be fulfilled.

exception claimstore.core.exception.**RestApiException**(*message*, *status_code=None*, *extra=None*)
Bases: Exception

Generic Rest API exception.

status_code = 400

Return exception as a dictionary.

claimstore.core.json module Useful JSON-related methods.

claimstore.core.json.**get_json_schema**(*schema*)
Return a given json schema.

Parameters **schema** (*str.*) – schema to be fetched. It must be a string with the format: module.schema_name (e.g. claims.claimants).

Returns a str with the requested json schema.

Return type str.

claimstore.core.json.**validate_json**(*json_input*, *schema*)
Validate JSON against a given schema.

Parameters

- **json_input** (*dict.*) – a dict with the full json to be validated.
- **schema** (*str.*) – JSON schema to use in the validation. It must be a string with the format module.schema_name (e.g. claims.claimants).

Raises `ValidationError` if the instance is invalid.

claimstore.core.pagination module Restful pagination.

class `claimstore.core.pagination.`**`RestfulSQLAlchemyPaginationMixIn`**
Bases: `object`

Implement Restful pagination for SQLAlchemy model and Flask-Restful.

It creates an instance of RequestParser that should be used by the Restful Resource implementation. By default, it adds two query fields to the Restful Resource:

Parameters

- **`page`** – page from which to fetch the data
- **`per_page`** – amout of data per page

Paginate query.

Parameters

- **`query`** – query object from SQLAlchemy.
- **`page`** – page from which to fetch data.
- **`per_page`** – amount of data per page.

Set Link details in the response header.

Parameters **`response`** – Flask Response object.

Module contents Core module for ClaimStore.

Submodules

claimstore.app module

Flask app creation.

`claimstore.app.`**`create_app`**`()`
Create Flask app using the factory.

`claimstore.app.`**`handle_restful_exceptions`**`(`*error*`)`
Handle invalid restful request exception.

claimstore.cli module

claimstore.config module

ClaimStore configuration.

claimstore.models module

ClaimStore data model.

class claimstore.models.**Claim**(***kwargs*)
Bases: flask_sqlalchemy.Model

Model representing a Claim.

Each claim is associated to a specific Claimant and references some already existing Identifier Types and predicate.

actor
Human that has performed the claim.

certainty
Certainty of the claim. It must be a float between 0 and 1.0.

claim_details
JSONB representation of the full claim as received.

claimant_id
Id of the associated Claimant.

created
Datetime in which the claim has been created by the claimant.

Get claims with the all the equivalent subjects or objects.

human
Whether the claims has been done by a human (1) or not (0).

id
Unique id of the claim.

object_eqid
Unique identifier for this object (type, value).

object_type_id
Id of the associated IdentifierType used as an object.

object_value
Value of the object.

predicate_id
Id of the associated Predicate.

received
Datetime in which the claim has been received and stored.

role
Role of the *human* who has performed the claim.

subject_eqid
Unique identifier for this subject (type, value).

subject_type_id
Id of the associated IdentifierType used as a subject.

subject_value
Value of the subject.

uuid
Universally Unique Identifier that represents a single claim.

class claimstore.models.**Claimant**(***kwargs*)
Bases: flask_sqlalchemy.Model

Represents a Claimant.

Claimants are the main providers of claims. Each claimant may be associated to many different claims.

claim
Claim associated with this claimant.

id
Unique id of the claimant.

joined
Datetime when the claimant subscribed to ClaimStore.

name
Claimant name. It must be unique and preferably short.

url
URL of the claimant.

uuid
Universally unique id of the claimant.

class claimstore.models.**EquivalentIdentifier**(***kwargs*)
Bases: flask_sqlalchemy.Model

Model that defines equivalent identifiers.

A given tuple (IdentifierType1, value1), e.g. (DOI, 1234) will have associated a unique id (*eqid*) that will be shared by the other tuple, e.g. (IdentifierType2, value2) in the relationship, in case the two IDs are equivalent (e.g. related via is_same_as predicate). If new equality claims are done in which existing identifiers are used, they will take the *eqid* from them.

For instance, if we have some claims like:

SubjectType	SubjectValue	Predicate	ObjectType	ObjectValue
Type1	Value1	is_same_as	Type2	Value2
Type3	Value3	is_same_as	Type4	Value4
Type1	Value1	is_same_as	Type4	Value4
Type5	Value5	is_same_as	Type6	Value6

Then, in the EquivalentIdentifier table there will be something like:

type	value	uuid
Type1	Value1	01
Type2	Value2	01
Type3	Value3	01
Type4	Value4	01
Type5	Value4	02
Type6	Value4	02

This table will enable and facilitate several use cases:

1. We could very easily get a list of the different identifiers for the same data resource.
2. It will simplify the recursive query by type/value in any subject/object claim.

Delete all the entries of the table equivalent_identifiers.

eqid
Universally Unique Identifier that represents a single data resource.

Return the identifiers of all the equivalent entities.

This method fetches the eqid for a given (type_name, value) and uses it to find all the equivalent identifiers. It returns a list with *EquivalentIdentifier.id*.

Return all the equivalent identifiers.

This method fetches the eqid for a given (type_name, value) and uses it to find all the equivalent identifiers.

id
Unique id of the data resource.

Rebuild index based on claims.

Store and return the equivalent identifiers as required.

type_id
The id of a given IdentifierType.

value
A given value for the IdentifierType.

class claimstore.models.**IdentifierType**(***kwargs*)
Bases: flask_sqlalchemy.Model

Represents an identifier type.

An Identifier Type is a persistent identifier type that can be used in claims by claimants.

claimant_id
Id of the associated claimant that registered this identifier.

description
Description of the identifier type.

eqid
Backref in equivalent_identifier to reach this identifier_type.

example_url
Example of a full URL in which the identifier is being used.

example_value
Example of a possible value for an identifier.

id
Unique id of the Identifier.

name
Unique name of the identifier type. Preferably one word in caps.

object
Backref in claim to reach this identifier_type.

subject
Backref in claim to reach this identifier_type.

url
URL in which the identifier is used.

class claimstore.models.**Predicate**(***kwargs*)
Bases: flask_sqlalchemy.Model

Represents a predicate.

The predicate defines the type of claim. An example of predicate could be: is_same_as.

claim
Backref in claim to reach this predicate.

description
Description of the predicate.

id
Unique id of the predicate.

name
Unique name of a predicate.

claimstore.restful module

Restful resources for the claims module.

class claimstore.restful.**ClaimResource**
Bases: *claimstore.restful.ClaimStoreResource*, *claimstore.core.pagination.RestfulSQLAlchemy*

Resource that handles all claims-related requests.

endpoint = 'claims'

GET service that returns the stored claims.

GET /api/claims/ (**uuid:** *claim_id*)
Returns a JSON list with all the claims matching the query parameters.

Request:

```
GET /api/claims?type=CDS_RECORD_ID&value=cond-mat/9906097&
recurse=1 HTTP/1.1
Accept: */*
Host: localhost:5000
```

```
GET /api/claims/0e64606e-68ce-482e-ad59-1e9981394f HTTP/1.1
Accept: */*
Host: localhost:5000
```

Request Headers

- Content-Type – application/json

Query Parameters

- **since** (*datetime*) – it must have the format 'YYYY-MM-DD'. It fetches claims that were created from this given datetime.
- **until** (*datetime*) – it must have the format 'YYYY-MM-DD'. It fetches claims that were created up to this given datetime.
- **claimant** (*string*) – claimant's unique name. It fetches claims submitted by the specified claimant.
- **predicate** (*string*) – predicate's unique name. It finds claims using this predicate (e.g. is_same_as).
- **certainty** (*float*) – float number between 0 and 1.0. It will search for claims with at least the specified certainty.
- **human** (*int*) – enter 1 if searching for human-created claims, 0 for algorithms and nothing in order to retrieve all.
- **actor** (*string*) – it filters claims by their actor's name (one can use %).
- **role** (*string*) – it filters claims by their actor's role (one can use %).

- **type** (*string*) – it finds claims using a certain identifier type (either subject or object). For instance: DOI.
- **value** (*string*) – it fetches all the claims with that identifier value.
- **recurse** (*boolean*) – used in combination with *type* and *value* will find all the equivalent identifiers to the specified one.
- **subject** (*string*) – it fetches claims using the given identifier type as a subject type.
- **object** (*string*) – it fetches claims using the given identifier type as an object type.
- **page** (*int*) – page from which to fetch data.
- **per_page** (*int*) – amount of data per page.

Response:

```
HTTP/1.0 200 OK
Content-Length: 1166
Content-Type: application/json

[
    {
        "arguments": {
            "actor": "CDS_submission",
            "human": 0
        },
        "certainty": 1.0,
        "claimant": "CDS",
        "created": "2015-03-25T11:00:00Z",
        "object": {
            "type": "CDS_REPORT_NUMBER",
            "value": "CMS-PAS-HIG-14-008"
        },
        "predicate": "is_same_as",
        "recieved": "2015-09-22T08:18:30.606912+00:00",
        "subject": {
            "type": "CDS_RECORD_ID",
            "value": "2003192"
        },
        "uuid": "44103ee2-0d87-47f9-b0e4-77673d297cdb"
    },
    {
        "arguments": {
            "actor": "John Doe",
            "human": 1,
            "role": "cataloguer"
        },
        "certainty": 0.5,
        "claimant": "INSPIRE",
        "created": "2015-05-25T11:00:00Z",
        "object": {
            "type": "CDS_RECORD_ID",
            "value": "2003192"
        },
        "predicate": "is_variant_of",
        "recieved": "2015-09-22T08:18:30.638933+00:00",
        "subject": {
            "type": "INSPIRE_RECORD_ID",
```

```
                    "value": "cond-mat/9906097"
                },
                "uuid": "27689445-02b9-4d5d-8f9b-da21970e2352"
            }
        ]
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – no error
- 400 Bad Request – invalid request
- 403 Forbidden – access denied

json_schema = 'claims.claim'

methods = ['GET', 'POST']

Record a new claim.

POST /api/claims

This resource is expecting JSON data with all the necessary information of a new claim.

Request:

```
POST /api/claims HTTP/1.1
Accept: application/json
Content-Length: 336
Content-Type: application/json

{
    "arguments": {
        "actor": "CDS_submission",
        "human": 0
    },
    "certainty": 1.0,
    "claimant": "CDS",
    "created": "2015-03-25T11:00:00Z",
    "object": {
        "type": "CDS_REPORT_NUMBER",
        "value": "CMS-PAS-HIG-14-008"
    },
    "predicate": "is_same_as",
    "subject": {
        "type": "CDS_RECORD_ID",
        "value": "2003192"
    }
}
```

Request Headers

- Content-Type – application/json

JSON Parameters

- **body** – JSON with the information of the claimt. The JSON data should be valid according to the JSON Schema for claims.

Responses:

```
HTTP/1.0 200 OK
Content-Length: 80
Content-Type: application/json

{
    "status": "success",
    "uuid": "fad4ec9f-0e95-4a22-b65c-d01f15aba6be"
}
```

```
HTTP/1.0 400 BAD REQUEST
Content-Length: 9616
Content-Type: application/json
Date: Tue, 22 Sep 2015 09:02:25 GMT
Server: Werkzeug/0.10.4 Python/3.4.3

{
    "extra": "'claimant' is a required property. Failed
              validating 'required' in schema...",
    "message": "JSON data is not valid",
    "status": 400
}
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – no error - the claim was recorded
- 400 Bad Request – invalid request - problably a malformed JSON
- 403 Forbidden – access denied

class claimstore.restful.**ClaimStoreResource**

Bases: flask_restful.Resource

Base class for REST resources.

json_schema = None

method_decorators = [<function error_handler at 0x7f7a2c4d1950>, <function check_ip at 0x7f7a2c4d19d8>]

Validate that json_data follows the appropiate JSON schema.

Parameters **json_data** – JSON data to be validated.

Raises InvalidJSONData if the instance is invalid.

class claimstore.restful.**ClaimantResource**

Bases: *claimstore.restful.ClaimStoreResource*

Resource related to claimant subscription in the ClaimStore.

This POST service expects JSON data following the JSON schema defined for claimants.

endpoint = 'claimants'

GET service that returns all the available claimants.

GET /api/claimants/(**uuid:** *claimant_id*)

Returns a JSON list with all the claimants.

Request:

```
GET /api/claimants HTTP/1.1
Accept: */*
Host: localhost:5000
```

```
GET /api/claimants/0e64606e-68ce-482e-ad59-1e99... HTTP/1.1
Accept: */*
Host: localhost:5000
```

Request Headers

- Content-Type – application/json

Response:

```
HTTP/1.0 200 OK
Content-Length: 108
Content-Type: application/json

[
    {
        "name": "ADS",
        "url": "http://adsabs.harvard.edu/",
        "uuid": "00000000-be54-45c0-89fa-00000000"
    },
    {
        "name": "ARXIV",
        "url": "http://arxiv.org/",
        "uuid": "00000000-87a6-4095-99a9-00000000"
    },
    {
        "name": "CDS",
        "url": "http://cds.cern.ch",
        "uuid": "00000000-602e-4912-8a9b-00000000"
    },
    {
        "name": "INSPIRE",
        "url": "http://inspirehep.net",
        "uuid": "00000000-42a5-4d3f-b54a-00000000"
    }
]
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – no error
- 400 Bad Request – invalid request
- 403 Forbidden – access denied

json_schema = 'claims.claimant'

methods = ['GET', 'POST']

Register a new claimant in the ClaimStore.

POST /api/claimants

This resource is expecting JSON data with all the necessary information of a new claimant.

Request:

```
POST /api/claimants HTTP/1.1
Content-Type: application/json
Host: localhost:5000

{
    "name": "INSPIRE",
    "url": "http://inspirehep.net"
}
```

Request Headers

- Content-Type – application/json

JSON Parameters

- **body** – JSON with the information of the claimant. The JSON data should be valid according to the JSON Schema for claimants.

Responses:

```
HTTP/1.0 200 OK
Content-Length: 80
Content-Type: application/json

{
    "status": "success",
    "uuid": "ab19c98b-xxxx-xxxx-xxxx-1d6af3bf58b4"
}
```

```
HTTP/1.0 400 BAD REQUEST
Content-Length: 95
Content-Type: application/json

{
    "extra": null,
    "message": "This claimant is already registered",
    "status": 400
}
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – no error - the claimant was subscribed
- 400 Bad Request – invalid request - problably a malformed JSON
- 403 Forbidden – access denied

class claimstore.restful.**EquivalentIdResource**

Bases: *claimstore.restful.ClaimStoreResource*

Resource that handles Equivalent Identifier requests.

endpoint = 'eqids'

GET service that returns all the stored Equivalent Identifiers.

GET /api/eqids/ (**uuid:** *eqid*)

Returns all the type/value entries in the index grouped by their equivalent identifiers.

Requests:

```
GET /api/eqids HTTP/1.1
Accept: */*
Host: localhost:5000
```

```
GET /api/eqids/0e64606e-68ce-482e-ad59-1e9981394f8 HTTP/1.1
Accept: */*
Host: localhost:5000
```

Request Headers

- Content-Type – application/json

Parameters

- **eqid** – query by a specific uuid which is shared by some equivalent identifiers

Response:

```
HTTP/1.0 200 OK
Content-Length: 592
Content-Type: application/json

{
    "36dfb125-5c35-4d3a-870c-76eb4bad498e": [
        {
            "type": "ARXIV_ID",
            "value": "cond-mat/9906097"
        },
        {
            "type": "DOI",
            "value": "C10.1103/PhysRevE.62.7422"
        }
    ],
    "77c4a5eb-3ed8-4c80-ba0d-644d6bc397a3": [
        {
            "type": "CDS_RECORD_ID",
            "value": "2003192"
        },
        {
            "type": "CDS_REPORT_NUMBER",
            "value": "CMS-PAS-HIG-14-008"
        },
        {
            "type": "INSPIRE_RECORD_ID",
            "value": "cond-mat/9906097"
        }
    ]
}
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – no error
- 400 Bad Request – invalid request - problably a malformed UUID
- 403 Forbidden – access denied

methods = ['GET']

class claimstore.restful.**IdentifierResource**
Bases: *claimstore.restful.ClaimStoreResource*

Resource that handles Identifier requests.

endpoint = 'identifiers'

GET service that returns the stored identifiers.

GET /api/identifiers
Returns a JSON list with all the available identifiers.

Request:

```
GET /api/identifiers HTTP/1.1
Accept: */*
Host: localhost:5000
```

Request Headers

- Content-Type – application/json

Response:

```
HTTP/1.0 200 OK
Content-Length: 147
Content-Type: application/json

[
    "ARXIV_ID",
    "CDS_AUTHOR_ID",
    "CDS_RECORD_ID",
    "CDS_REPORT_NUMBER",
    "DOI",
    "INSPIRE_AUTHOR_ID",
    "INSPIRE_RECORD_ID"
]
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – no error
- 400 Bad Request – invalid request
- 403 Forbidden – access denied

methods = ['GET']

class claimstore.restful.**PredicateResource**
Bases: *claimstore.restful.ClaimStoreResource*

Resource that handles Predicate requests.

endpoint = 'predicates'

GET service that returns all the available predicates.

GET /api/predicates
Returns a JSON list with all the predicates.

Request:

```
GET /api/predicates HTTP/1.1
Accept: */*
Host: localhost:5000
```

Request Headers

- Content-Type – application/json

Response:

```
HTTP/1.0 200 OK
Content-Length: 108
Content-Type: application/json

[
    "is_author_of",
    "is_contributor_to",
    "is_erratum_of",
    "is_same_as",
    "is_variant_of"
]
```

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – no error
- 400 Bad Request – invalid request
- 403 Forbidden – access denied

methods = ['GET']

claimstore.restful.**check_ip**(*f*)
Decorator to control the access to the API.

If the client's IP matches the list of IPs defined in the environment variable *CLAIMSTORE_ALLOWED_IPS*, then the access will be granted. Otherwise, an access denied code 403 will be raised.

claimstore.restful.**error_handler**(*f*)
Decorator to handle restful exceptions.

If this decorator is not used, Flask-RestFul will always raise a 500 code, independently of your Flask app error handler registration.

claimstore.version module

ClaimStore version number.

claimstore.views module

ClaimStore views.

`claimstore.views.`**`claimsubmit`**`()`
Render the claim submission form page.

`claimstore.views.`**`contact`**`()`
Render the contact page.

`claimstore.views.`**`index`**`()`
Render the home page for ClaimStore.

`claimstore.views.`**`subscription`**`()`
Render the subscription form page.

claimstore.wsgi module

WSGI app creation.

Module contents

Main claimstore module.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

C

/api