plone.restapi Documentation

Release 1.0a1

Plone Foundation

May 22, 2016

Warning: plone.restapi ist still in a pre-alpha stage. Any functionality described by this document might not be currently available on Plone, and/or might change significantly in the future.

CHAPTER 1

Contents

1.1 Introduction

A hypermedia API provides an entry point to the API, which contains hyperlinks the clients can follow. Just like a human user of a regular website, who knows the initial URL of a website and then follows hyperlinks to navigate through the site. This has the advantage that the client only needs to understand how to detect and follow links. The URLs (apart from the inital entry point) and other details of the API can change without breaking the client.

The entry point to the Plone RESTful API is the portal root. The client can ask for a *REST* API response by setting the `'Accept'` HTTP header to `'application/json'`:

```
GET /
Accept: application/json
```

This uses so-called 'content negotiation'

1.1.1 Content Negotiation

Content negotiation is a mechanism defined in the HTTP specification that makes it possible to serve different versions of a document (or more generally, a resource representation) at the same URI, so that user agents can specify which version fit their capabilities the best. The user agent (or the REST consumer) can ask for a specific representation by providing an Accept HTTP header that lists acceptable media types (e.g. JSON):

```
GET /
Accept: application/json
```

The server is then able to supply the version of the resource that best fits the user agent's needs. This is reflected in the Content-Type header:

```
HTTP 200 OK
Content-Type: application/json

{
  'data': ...
}
```

The server will then respond with the portal root in the JSON format:

```
GET /plone
Accept: application/json
```

```
HTTP 200 OK
content-type: application/json

{
  "@id": "http://localhost:55001/plone",
  "@type": "Plone Site",
  "member": [
    {
      "@id": "http://localhost:55001/plone/robot-test-folder",
      "@type": "Folder",
      "description": "",
      "title": "Test Folder"
    },
    {
      "@id": "http://localhost:55001/plone/front-page",
      "@type": "Document",
      "description": "Congratulations! You have successfully installed Plone.",
      "title": "Welcome to Plone"
    }
  ],
  "parent": {}
}
```

@id is a unique identifier for resources (IRIs). The *@id* property can be used to navigate through the web API by following the links.

@type sets the data type of a node or typed value

member is a list that contains all objects within that resource.

A client application can "follow" the links (by calling the @id property) to other resources. This allows to build a losely coupled client that does not break if some of the URLs change, only the entry point of the entire API (in our case the portal root) needs to be known in advance.

Another example, this time showing syntax in curl, as an http-request and how a python request would look like. Click on the buttons below to show the different syntaxes for the request.

```
curl -i -H "Accept: application/json" -X GET http://localhost:8080/Plone/document
```

```
http GET http://localhost:8080/Plone/document Accept:application/json
```

```
requests.get('http://localhost:8080/Plone/document', auth=('admin', 'admin'), headers={'Accept': 'app
```

```
GET /plone/front-page
Accept: application/json

HTTP 200 OK
content-type: application/json

{
  "@id": "http://localhost:55001/plone/front-page",
  "@type": "Document",
  "UID": "1f699ffa110e45afb1ba502f75f7ec33",
  "allow_discussion": null,
  "changeNote": "",
  "contributors": [],
  "created": "2016-01-21T01:14:48+00:00",
  "creators": [
    "test_user_1_",
    "admin"
  ],
  "description": "Congratulations! You have successfully installed Plone.",
  "effective": null,
  "exclude_from_nav": false,
  "expires": null,
  "language": "",
  "modified": "2016-01-21T01:24:11+00:00",
  "parent": {
    "@id": "http://localhost:55001/plone",
    "@type": "Plone Site",
    "description": "",
    "title": "Plone site"
  },
  "relatedItems": [],
  "rights": "",
  "subjects": [],
  "table_of_contents": null,
  "text": {
    "content-type": "text/plain",
    "data": "If you're seeing this instead of the web site you were expecting, the owner of this web
    "encoding": "utf-8"
  },
  "title": "Welcome to Plone"
}
```

And so on, see

1.1.2 Plone Content

How to get all standard Plone content representations. The syntax is given in various tools, click on ‘curl’, ‘http-request’ or ‘python-requests’ to see examples. **Plone Portal Root:**

```
curl -i -H "Accept: application/json" -X GET http://localhost:8080/Plone
```

```
http GET http://localhost:8080/Plone Accept:application/json
```

```
requests.get('http://localhost:8080/Plone', auth=('admin', 'admin'), headers={'Accept': 'applica
```

```
GET /plone
Accept: application/json

HTTP 200 OK
content-type: application/json

{
  "@id": "http://localhost:55001/plone",
  "@type": "Plone Site",
  "member": [
    {
      "@id": "http://localhost:55001/plone/robot-test-folder",
      "@type": "Folder",
      "description": "",
      "title": "Test Folder"
    },
    {
      "@id": "http://localhost:55001/plone/front-page",
      "@type": "Document",
      "description": "Congratulations! You have successfully installed Plone.",
      "title": "Welcome to Plone"
    }
  ],
  "parent": {}
}
```

Plone Folder:

```
curl -i -H "Accept: application/json" -X GET http://localhost:8080/Plone/folder
```

```
http GET http://localhost:8080/Plone/folder Accept:application/json
```

```
requests.get('http://localhost:8080/Plone/folder', auth=('admin', 'admin'), headers={'Accept': '
```

```
GET /plone/folder
Accept: application/json

HTTP 200 OK
content-type: application/json

{
  "@id": "http://localhost:55001/plone/folder",
  "@type": "Folder",
  "UID": "fc7881c46d61452db4177bc059d9dcb5",
  "allow_discussion": null,
  "contributors": [],
  "created": "2016-01-21T07:14:48+00:00",
  "creators": [
    "test_user_1_",
    "admin"
  ],
  "description": "This is a folder with two documents",
  "effective": null,
  "exclude_from_nav": false,
  "expires": null,
  "language": "",
  "member": [
    {
      "@id": "http://localhost:55001/plone/folder/doc1",
      "@type": "Document",
      "description": "",
      "title": "A document within a folder"
    },
    {
      "@id": "http://localhost:55001/plone/folder/doc2",
      "@type": "Document",
      "description": "",
      "title": "A document within a folder"
    }
  ],
  "modified": "2016-01-21T07:24:11+00:00",
  "nextPreviousEnabled": false,
  "parent": {
    "@id": "http://localhost:55001/plone",
    "@type": "Plone Site",
    "description": "",
    "title": "Plone site"
  },
  "relatedItems": [],
  "rights": "",
  "subjects": [],
  "title": "My Folder"
}
```

Plone Document:

```
curl -i -H "Accept: application/json" -X GET http://localhost:8080/Plone/document
```

```
http GET http://localhost:8080/Plone/document Accept:application/json
```

```
requests.get('http://localhost:8080/Plone/document', auth=('admin', 'admin'), headers={'Accept':
```

```
GET /plone/front-page
Accept: application/json

HTTP 200 OK
content-type: application/json

{
  "@id": "http://localhost:55001/plone/front-page",
  "@type": "Document",
  "UID": "1f699ffa110e45afb1ba502f75f7ec33",
  "allow_discussion": null,
  "changeNote": "",
  "contributors": [],
  "created": "2016-01-21T01:14:48+00:00",
  "creators": [
    "test_user_1_",
    "admin"
  ],
  "description": "Congratulations! You have successfully installed Plone.",
  "effective": null,
  "exclude_from_nav": false,
  "expires": null,
  "language": "",
  "modified": "2016-01-21T01:24:11+00:00",
  "parent": {
    "@id": "http://localhost:55001/plone",
    "@type": "Plone Site",
    "description": "",
    "title": "Plone site"
  },
  "relatedItems": [],
  "rights": "",
  "subjects": [],
  "table_of_contents": null,
  "text": {
    "content-type": "text/plain",
    "data": "If you're seeing this instead of the web site you were expecting, the owner of this
    "encoding": "utf-8"
  },
  "title": "Welcome to Plone"
}
```

News Item:

```
curl -i -H "Accept: application/json" -X GET http://localhost:8080/Plone/newsitem
```

```
http GET http://localhost:8080/Plone/newsitem Accept:application/json
```

```
requests.get('http://localhost:8080/Plone/newsitem', auth=('admin', 'admin'), headers={'Accept':
```

```
GET /plone/newsitem
Accept: application/json

HTTP 200 OK
content-type: application/json

{
  "@id": "http://localhost:55001/plone/newsitem",
  "@type": "News Item",
  "UID": "80c2a074cb4240d08c9a129e3a834c74",
  "allow_discussion": null,
  "changeNote": "",
  "contributors": [],
  "created": "2016-01-21T02:14:48+00:00",
  "creators": [
    "test_user_1_",
    "admin"
  ],
  "description": "This is a news item",
  "effective": null,
  "exclude_from_nav": false,
  "expires": null,
  "image": {
    "icon": "http://localhost:55001/plone/newsitem/@@images/image/icon",
    "large": "http://localhost:55001/plone/newsitem/@@images/image/large",
    "listing": "http://localhost:55001/plone/newsitem/@@images/image/listing",
    "mini": "http://localhost:55001/plone/newsitem/@@images/image/mini",
    "original": "http://localhost:55001/plone/newsitem/@@images/image",
    "preview": "http://localhost:55001/plone/newsitem/@@images/image/preview",
    "thumb": "http://localhost:55001/plone/newsitem/@@images/image/thumb",
    "tile": "http://localhost:55001/plone/newsitem/@@images/image/tile"
  },
  "image_caption": "This is an image caption.",
  "language": "",
  "modified": "2016-01-21T02:24:11+00:00",
  "parent": {
    "@id": "http://localhost:55001/plone",
    "@type": "Plone Site",
    "description": "",
    "title": "Plone site"
  },
  "relatedItems": [],
  "rights": "",
  "subjects": [],
  "text": {
    "content-type": "text/plain",
    "data": "Lorem ipsum",
    "encoding": "utf-8"
  },
  "title": "My News Item"
}
```

Event:

```
curl -i -H "Accept: application/json" -X GET http://localhost:8080/Plone/event
```

```
http GET http://localhost:8080/Plone/event Accept:application/json
```

```
requests.get('http://localhost:8080/Plone/event', auth=('admin', 'admin'), headers={'Accept': 'a
```

```
GET /plone/event
Accept: application/json

HTTP 200 OK
content-type: application/json

{
  "@id": "http://localhost:55001/plone/event",
  "@type": "Event",
  "UID": "846d632bc0854c5aa6d3dcae171ed2db",
  "allow_discussion": null,
  "attendees": [],
  "changeNote": "",
  "contact_email": null,
  "contact_name": null,
  "contact_phone": null,
  "contributors": [],
  "created": "2016-01-21T03:14:48+00:00",
  "creators": [
    "test_user_1_",
    "admin"
  ],
  "description": "This is an event",
  "effective": null,
  "end": null,
  "event_url": null,
  "exclude_from_nav": false,
  "expires": null,
  "language": "",
  "location": null,
  "modified": "2016-01-21T03:24:11+00:00",
  "open_end": null,
  "parent": {
    "@id": "http://localhost:55001/plone",
    "@type": "Plone Site",
    "description": "",
    "title": "Plone site"
  },
  "recurrence": null,
  "relatedItems": [],
  "rights": "",
  "start": null,
  "subjects": [],
  "sync_uid": null,
  "text": null,
  "timezone": null,
  "title": "Event",
  "whole_day": null
}
```

Image:

```
curl -i -H "Accept: application/json" -X GET http://localhost:8080/Plone/image
```

```
http GET http://localhost:8080/Plone/image Accept:application/json
```

```
requests.get('http://localhost:8080/Plone/image', auth=('admin', 'admin'), headers={'Accept': 'a
```

```
GET /plone/image
Accept: application/json

HTTP 200 OK
content-type: application/json

{
  "@id": "http://localhost:55001/plone/image",
  "@type": "Image",
  "UID": "2166e81a0c224fe3b62e197c7fdc9c3e",
  "allow_discussion": null,
  "contributors": [],
  "created": "2016-01-21T06:14:48+00:00",
  "creators": [
    "test_user_1_",
    "admin"
  ],
  "description": "This is an image",
  "effective": null,
  "exclude_from_nav": false,
  "expires": null,
  "image": {
    "icon": "http://localhost:55001/plone/image/@@images/image/icon",
    "large": "http://localhost:55001/plone/image/@@images/image/large",
    "listing": "http://localhost:55001/plone/image/@@images/image/listing",
    "mini": "http://localhost:55001/plone/image/@@images/image/mini",
    "original": "http://localhost:55001/plone/image/@@images/image",
    "preview": "http://localhost:55001/plone/image/@@images/image/preview",
    "thumb": "http://localhost:55001/plone/image/@@images/image/thumb",
    "tile": "http://localhost:55001/plone/image/@@images/image/tile"
  },
  "language": "",
  "modified": "2016-01-21T06:24:11+00:00",
  "parent": {
    "@id": "http://localhost:55001/plone",
    "@type": "Plone Site",
    "description": "",
    "title": "Plone site"
  },
  "relatedItems": [],
  "rights": "",
  "subjects": [],
  "title": "My Image"
}
```

File:

```
curl -i -H "Accept: application/json" -X GET http://localhost:8080/Plone/file
```

```
http GET http://localhost:8080/Plone/file Accept:application/json
```

```
requests.get('http://localhost:8080/Plone/file', auth=('admin', 'admin'), headers={'Accept': 'ap
```

```
GET /plone/file
Accept: application/json

HTTP 200 OK
content-type: application/json

{
  "@id": "http://localhost:55001/plone/file",
  "@type": "File",
  "UID": "9b6a4eadb9074dde97d86171bb332ae9",
  "allow_discussion": null,
  "contributors": [],
  "created": "2016-01-21T05:14:48+00:00",
  "creators": [
    "test_user_1_",
    "admin"
  ],
  "description": "This is a file",
  "effective": null,
  "exclude_from_nav": false,
  "expires": null,
  "file": "http://localhost:55001/plone/file/@@download/file",
  "language": "",
  "modified": "2016-01-21T05:24:11+00:00",
  "parent": {
    "@id": "http://localhost:55001/plone",
    "@type": "Plone Site",
    "description": "",
    "title": "Plone site"
  },
  "relatedItems": [],
  "rights": "",
  "subjects": [],
  "title": "My File"
}
```

Link:

```
curl -i -H "Accept: application/json" -X GET http://localhost:8080/Plone/link
```

```
http GET http://localhost:8080/Plone/link Accept:application/json
```

```
requests.get('http://localhost:8080/Plone/link', auth=('admin', 'admin'), headers={'Accept': 'ap
GET /plone/link
Accept: application/json

HTTP 200 OK
content-type: application/json

{
  "@id": "http://localhost:55001/plone/link",
  "@type": "Link",
  "UID": "6ff48d27762143a0ae8d63cee73d9fc2",
  "allow_discussion": null,
  "changeNote": "",
  "contributors": [],
  "created": "2016-01-21T04:14:48+00:00",
  "creators": [
    "test_user_1_",
    "admin"
  ],
  "description": "This is a link",
  "effective": null,
  "exclude_from_nav": false,
  "expires": null,
  "language": "",
  "modified": "2016-01-21T04:24:11+00:00",
  "parent": {
    "@id": "http://localhost:55001/plone",
    "@type": "Plone Site",
    "description": "",
    "title": "Plone site"
  },
  "remoteUrl": "http://",
  "rights": "",
  "subjects": [],
  "title": "My Link"
}
```

Collection:

```
curl -i -H "Accept: application/json" -X GET http://localhost:8080/Plone/collection
```

```
http GET http://localhost:8080/Plone/collection Accept:application/json
```

```
requests.get('http://localhost:8080/Plone/collection', auth=('admin', 'admin'), headers={'Accept
```

```
GET /plone/collection
Accept: application/json

HTTP 200 OK
content-type: application/json

{
  "@id": "http://localhost:55001/plone/collection",
  "@type": "Collection",
  "UID": "d0c89bc77f874ce1aad5720921d875c0",
  "allow_discussion": null,
  "contributors": [],
  "created": "2016-01-21T08:14:48+00:00",
  "creators": [
    "test_user_1_",
    "admin"
  ],
  "customViewFields": [],
  "description": "This is a collection with two documents",
  "effective": null,
  "exclude_from_nav": false,
  "expires": null,
  "item_count": 30,
  "language": "",
  "limit": 1000,
  "member": [
    {
      "@id": "http://localhost:55001/plone/front-page",
      "@type": "Document",
      "description": "Congratulations! You have successfully installed Plone.",
      "title": "Welcome to Plone"
    },
    {
      "@id": "http://localhost:55001/plone/doc1",
      "@type": "Document",
      "description": "",
      "title": "Document 1"
    },
    {
      "@id": "http://localhost:55001/plone/doc2",
      "@type": "Document",
      "description": "",
      "title": "Document 2"
    }
  ],
  "modified": "2016-01-21T08:24:11+00:00",
  "parent": {
    "@id": "http://localhost:55001/plone",
    "@type": "Plone Site",
    "description": "",
    "title": "Plone site"
  },
  "query": [
    {
      "i": "portal_type",
      "o": "plone.app.querystring.operation.string.is",
      "v": "Document"
    }
  ],
  "relatedItems": [],
  "rights": "",
  "sort_on": null,
  "sort_reversed": null,
  "subjects": [],
  "text": null,
  "title": "My Collection"
}
```

1.2 Authentication

`plone.restapi` uses Plone PAS for Authentication.

That means that any authentication method supported by an installed PAS Plugin should work, assuming it's an authentication method that makes sense to use with an API.

For example, to authenticate using HTTP basic auth, you'd set an `Authorization` header:

```
GET /Plone HTTP/1.1
Authorization: Basic Zm9vYmFyOmZvb2Jhcgo=
Accept: application/json
```

HTTP client libraries usually contain helper functions to produce a proper `Authorization` header for you based on given credentials.

Using the `requests` library, you'd set up a session with basic auth like this:

```python
import requests

session = requests.Session()
session.auth = ('username', 'password')
session.headers.update({'Accept': 'application/json'})

response = session.get(url)
```

Or the same example using `curl`:

```
curl -u username:password -H 'Accept:application/json' $URL
```

1.3 CRUD Web Services

CRUD is a pattern for manipulating resources across the network by using HTTP as an application protocol. The CRUD operations (Create, Read, Update, Delete) can be mapped to the corresponding HTTP verbs POST (Create), GET (Read), PUT (Update) and DELETE (Delete). This allows us to interact with a specific resource in a standardized way:

Verb	URL	Action
POST	/folder	Creates a new document within the folder
GET	/folder/{documentId}	Request the current state of the document
PUT	/folder/{documentId}	Update the document details
DELETE	/folder/{documentId}	Remove the document

1.3.1 Creating a Resource with POST

To create a new resource, we send a POST request to the resource container. If we want to create a new document within an existing folder, we send a POST request to that folder:

```
POST /folder HTTP/1.1
Host: localhost:8080
Accept: application/json
Content-Type: application/json

{
    '@type': 'Document',
    'title': 'My Document',
}
```

```
curl -i -H "Accept: application/json" -H "Content-type: application/json" --data-raw '{"@type":"Docum
```

```
http -a admin:admin POST http://localhost:8080/Plone/folder \\@type=Document title=My Document Accept

.. code-block:: python-requests

  requests.post('http://localhost:8080/Plone/folder', auth=('admin', 'admin'), headers={'Accept': 'ap
```

By setting the 'Accept' header, we tell the server that we would like to receive the response in the 'application/json' representation format.

The 'Content-Type' header indicates that the body uses the 'application/json' format.

The request body contains the minimal necessary information needed to create a document (the type and the title). You could set other properties, like "description" here as well.

Successful Response (201 Created)

If a resource has been created, the server responds with the *201 Created* status code. The 'Location' header contains the URL of the newly created resource and the resource represenation in the payload:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: http://localhost:8080/folder/my-document

{
    '@type': 'Document',
    'id': 'my-document',
    'title': 'My Document',
}
```

Unsuccessful Response (400 Bad Request)

If the resource could not be created, for instance because the title was missing in the request, the server responds with *400 Bad Request*:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
  'message': 'Required title field is missing'
}
```

The response body can contain information about why the request failed.

Unsuccessful Response (500 Internal Server Error)

If the server can not properly process a request, it responds with *500 Internal Server Error*:

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json

{
  'message': 'Internal Server Error'
}
```

The response body can contain further information such as an error trace or a link to the documentation.

Possible POST Responses

Possible server reponses for a POST request are:

- *201 Created* (Resource has been created successfully)
- *400 Bad Request* (malformed request to the service)
- *500 Internal Server Error* (server fault, can not recover internally)

POST Implementation

A pseudo-code example of the POST implementation on the server:

```
try:
    order = createOrder()
    if order == None:
        # Bad Request
        response.setStatus(400)
    else:
        # Created
        response.setStatus(201)
except:
    # Internal Server Error
    response.setStatus(500)
```

TODO: Link to the real implementation...

1.3.2 Reading a Resource with GET

After a successful POST, we can access the resource by sending a GET request to the resource URL:

```
GET /folder/my-document HTTP/1.1
Host: localhost:8080
Accept: application/json
```

```
curl -i -H "Accept: application/json" --user admin:admin -X GET http://localhost:8080/Plone/folder/my
```

```
http -a admin:admin GET http://localhost:8080/Plone/folder/my-document Accept:application/json
```

```
requests.get('http://localhost:8080/Plone/folder/my-document', auth=('admin', 'admin'), headers={'Acc
```

Successful Response (200 OK)

If a resource has been retrieved successfully, the server responds with *200 OK*:

```
GET /plone/front-page
Accept: application/json

HTTP 200 OK
```

```
content-type: application/json

{
  "@id": "http://localhost:55001/plone/front-page",
  "@type": "Document",
  "UID": "1f699ffa110e45afb1ba502f75f7ec33",
  "allow_discussion": null,
  "changeNote": "",
  "contributors": [],
  "created": "2016-01-21T01:14:48+00:00",
  "creators": [
    "test_user_1_",
    "admin"
  ],
  "description": "Congratulations! You have successfully installed Plone.",
  "effective": null,
  "exclude_from_nav": false,
  "expires": null,
  "language": "",
  "modified": "2016-01-21T01:24:11+00:00",
  "parent": {
    "@id": "http://localhost:55001/plone",
    "@type": "Plone Site",
    "description": "",
    "title": "Plone site"
  },
  "relatedItems": [],
  "rights": "",
  "subjects": [],
  "table_of_contents": null,
  "text": {
    "content-type": "text/plain",
    "data": "If you're seeing this instead of the web site you were expecting, the owner of this web
    "encoding": "utf-8"
  },
  "title": "Welcome to Plone"
}
```

Unsuccessful response (404 Not Found)

If a resource could not be found, the server will respond with *404 Not Found*:

```
HTTP/1.1 404 Not Found
Content-Type: application/json

{
  'error': 'NotFound'
}
```

GET Implementation

A pseudo-code example of the GET implementation on the server:

```
try:
    order = getOrder()
    if order == None:
        # Not Found
        response.setStatus(404)
    else:
        # OK
        response.setStatus(200)
except:
    # Internal Server Error
    response.setStatus(500)
```

TODO: Link to the real implementation...

GET Responses

Possible server reponses for a GET request are:

- *200 OK*
- *404 Not Found*
- *500 Internal Server Error*

1.3.3 Updating a Resource with PATCH

To update an existing resource we send a PATCH request to the server. PATCH allows to provide just a subset of the resource (the values you actually want to change):

```
PATCH /folder/my-document HTTP/1.1
Host: localhost:8080/Plone
Content-Type: application/json

{
    '@type': 'Document',
    'title': 'My New Document Title',
}
```

```
curl -i -H "Accept: application/json" -H "Content-type: application/json" --data-raw '{"title": "My Do
```

```
http -a admin:admin PATCH http://localhost:8080/Plone/folder/my-document title="My New Document Title
```

```
requests.patch('http://localhost:8080/Plone/folder/my-document', auth=('admin', 'admin'), headers={'A
```

Successful Response (204 No Content)

A successful response to a PATCH request will be indicated by a *204 No Content* response:

```
HTTP/1.1  204 No Content
```

See for full specs the RFC 5789: PATCH Method for HTTP

1.3.4 Replacing a Resource with PUT

Note: PUT is not implemented yet.

To replace an existing resource we send a PUT request to the server:

```
PUT /folder/my-document HTTP/1.1
Host: localhost:8080
Content-Type: application/json

{
    '@type': 'Document',
    'title': 'My New Document Title',
}
```

```
curl -i -H "Accept: application/json" -X PUT http://localhost:8080/Plone/folder
```

```
http -a admin:admin PUT http://localhost:8080/Plone/folder title="My New Document Title" Accept:appl
```

```
requests.put('http://localhost:8080/Plone/folder/my-document', auth=('admin', 'admin'), headers={'Ac
```

In accordance with the HTTP specification, a successful PUT will not create a new resource or produce a new URL.

PUT expects the entire resource representation to be supplied to the server, rather than just changes to the resource state. This is usually not a problem since the consumer application requested the resource representation before a PUT anyways.

When the PUT request is accepted and processed by the service, the consumer will receive a *204 No Content* response (*200 OK* would be a valid alternative).

Successful Update (204 No Content)

When a resource has been updated successfully, the server sends a *204 No Content* response:

```
HTTP/1.1 204 No Content
Content-Type:: application/json
```

Unsuccessful Update (409 Conflict)

Sometimes requests fail due to incompatible changes. The response body includes additional information about the problem.

TODO: Add example.

PUT Implementation

A pseudo-code example of the PUT implementation on the server:

```
try:
    order = getOrder()
    if order:
        try:
            saveOrder()
        except conflict:
            response.setStatus(409)
        # OK
        response.setStatus(200)
    else:
        # Not Found
        response.setStatus(404)
```

```
except:
    # Internal Server Error
    response.setStatus(500)
```

TODO: Link to the real implementation...

PUT Responses

Possible server reponses for a PUT request are:

- *200 OK*
- *404 Not Found*
- *409 Conflict*
- *500 Internal Server Error*

POST vs. PUT

Difference between POST and PUT:

- Use POST to create a resource identified by a service-generated URI
- Use POST to append a resource to a collection identified by a service-generated URI
- Use PUT to overwrite a resource

This follows RFC 7231: HTTP 1.1: PUT Method.

1.3.5 Removing a Resource with DELETE

We can delete an existing resource by sending a DELETE request:

```
DELETE /folder/my-document HTTP/1.1
Host: localhost:8080
```

```
curl -i -H "Accept: application/json" -X DELETE --user admin:admin http://localhost:8080/Plone/folde
```

```
http -a admin:admin DELETE http://localhost:8080/Plone/folder/my-document Accept:application/json
```

```
requests.delete('http://localhost:8080/Plone/folder', auth=('admin', 'admin'), headers={'Accept': 'a
```

A successful response will be indicated by a *204 No Content* response:

```
HTTP/1.1  204 No Content
```

DELETE Implementation

A pseudo-code example of the DELETE implementation on the server:

```
try:
    order = getOrder()
    if order:
        if can_delete(order):
            # No Content
            response.setStatus(204)
        else:
            # Not Allowed
            response.setStatus(405)
    else:
        # Not Found
        response.setStatus(404)
except:
    # Internal Server Error
    response.setStatus(500)
```

TODO: Link to the real implementation...

DELETE Responses

Possible responses to a delete request are:

- *204 No Content*
- *404 Not Found* (if the resource does not exist)
- *405 Method Not Allowed* (if deleting the resource is not allowed)
- *500 Internal Server Error*

1.4 Workflow

Note: Currently the workflow support is limited to executing transitions on content.

In Plone, content almost always has a *workflow* attached. We can get the current state and history of an object by issuing a GET request using on any context:

```
GET /plone/front-page/@workflow
Accept: application/json

HTTP 200 OK
content-type: application/json

{
  "history": [
    {
      "action": null,
      "actor": "test_user_1_",
      "comments": "",
      "review_state": "private",
      "time": "2016-05-19T10:32:40+00:00"
    }
  ],
  "transitions": [
    {
      "@id": "http://localhost:55001/plone/front-page/@workflow/publish",
      "title": "Publish"
    },
    {
      "@id": "http://localhost:55001/plone/front-page/@workflow/submit",
      "title": "Submit for publication"
    }
  ]
}
```

Now, if we want to change the state of the front page to publish, we would proceed by issuing a `POST` request to the given URL:

```
POST /plone/front-page/@workflow/publish
Accept: application/json

HTTP 200 OK
content-type: application/json

{
  "action": "publish",
  "actor": "admin",
  "comments": "",
  "review_state": "published",
  "time": "2016-05-19T10:32:40+00:00"
}
```

1.5 Registry

Registry records can be addressed by using the fully qualified dotted name of the registry record to be read/written as the `:name` parameter.

Reading or writing registry records requires the `cmf.ManagePortal` permission.

1.5.1 Reading registry records

Reading a single record:

```
GET /:portal/@registry/:name HTTP/1.1
Host: localhost:8080
Accept: application/json
```

Example:

```
GET /plone/@registry/plone.app.querystring.field.path.title
Accept: application/json

HTTP 200 OK
content-type: application/json

"Location"
```

1.5.2 Updating registry records

Updating an existing record:

```
PATCH /:portal/@registry/ HTTP/1.1
Host: localhost:8080
Accept: application/json
```

```
{name: value}
```

Example:

```
PATCH /plone/@registry/
Accept: application/json

HTTP 204 No Content
```

1.6 Serialization

Throughout the REST API, content needs to be serialized and deserialized to and from JSON representations.

In general, the format used for serializing content when reading from the API is the same as is used to submit content to the API for writing.

1.6.1 Basic Types

Basic Python data types that have a corresponding type in JSON, like integers or strings, will simply be translated between the Python type and the respective JSON type.

1.6.2 Dates and Times

Since JSON doesn’t have native support for dates/times, the Python/Zope datetime types will be serialized to an ISO 8601 datestring.

Python	JSON
`time(19, 45, 55)`	`'19:45:55'`
`date(2015, 11, 23)`	`'2015-11-23'`
`datetime(2015, 11, 23, 19, 45, 55)`	`'2015-11-23T19:45:55'`
`DateTime('2015/11/23 19:45:55')`	`'2015-11-23T19:45:55'`

1.6.3 RichText fields

RichText fields will be serialized as follows:

A `RichTextValue` like

```
RichTextValue(u'<p>Hallöchen</p>',
              mimeType='text/html',
              outputMimeType='text/html')
```

will be serialized to

```
{
  "data": "<p>Hall\u00f6chen</p>",
  "content-type": "text/html",
  "encoding": "utf-8"
}
```

1.6.4 File Fields

Download (serialization)

For download, the file field will simply be serialized to a string that contains the download URL for the file.

```
{
  "...": "",
  "@type": "File",
  "title": "My file",
  "file": "http://localhost:55001/plone/file/@@download/file"
}
```

That URL points to the regular Plone download view.

This means that when accessing that URL, your request won't be handled by the API but a regular Plone browser view. Therefore you must **not** send the `Accept: application/json` header in this case.

Upload (deserialization)

For a field of type `Named[Blob]File` (Dexterity) or `FileField` (Archetypes), represent the field content as a dictionary with these four keys:

- `data` - the base64 encoded contents of the file
- `encoding` - the encoding you used to encode the data, so usually *base64*
- `content-type` - the MIME type of the file
- `filename` - the name of the file, including extension

```
{
  "...": "",
  "@type": "File",
  "title": "My file",
  "file": {
      "data": "TG9yZW0gSXBzdW0uCg==",
      "encoding": "base64",
      "filename": "lorem.txt",
      "content-type": "text/plain"}
}
```

1.6.5 Relations

Serialization

A `RelationValue` will be serialized to a short summary representation of the referenced object:

```
{
  '@id': 'http://nohost/plone/doc1',
  '@type': 'DXTestDocument',
  'title': 'Document 1',
  'description': 'Description'
}
```

The `RelationList` containing that reference will be represented as a list in JSON.

Deserialization

In order to set a relation when creating or updating content, you can use one of several ways to specify relations:

Type	Example
UID	`'9b6a4eadb9074dde97d86171bb332ae9'`
IntId	`123456`
Path	`'/plone/doc1'`
URL	`'http://localhost:8080/plone/doc1'`

1.7 Search

Content in a Plone site can be searched for by invoking the `/@search` endpoint on any context:

```
GET /plone/@search HTTP/1.1
Accept: application/json
```

A search is **contextual** by default, i.e. it is bound to a specific collection and searches within that collection and any sub-collections.

Since a Plone site is also a collection, we therefore have a global search (by invoking the `/@search` endpoint on the site root) and contextual searches (by invoking that endpoint on any other context) all using the same pattern.

In terms of the resulting catalog query this means that, by default, a search will be constrained by the path to the context it's invoked on, unless you explicitly supply your own `path` query.

Search results are represented similar to collections:

```
GET /plone/@search?sort_on=path
Accept: application/json

HTTP 200 OK
content-type: application/json

{
  "items_count": 2,
  "member": [
    {
      "@id": "http://localhost:55001/plone/front-page",
      "@type": "Document",
      "description": "Congratulations! You have successfully installed Plone.",
      "title": "Welcome to Plone"
    },
    {
      "@id": "http://localhost:55001/plone/robot-test-folder",
      "@type": "Folder",
      "description": "",
      "title": "Test Folder"
    }
  ]
}
```

The default representation for search results is a summary that contains only the most basic information. In order to return specific metadata columns, see the documentation of the `metadata_fields` parameter below.

1.7.1 Query format

Queries and query-wide options (like `sort_on`) are submitted as query string parameters to the `/@search` request:

```
GET /plone/@search?SearchableText=lorem
```

This is nearly identical to the way that queries are passed to the Plone `@@search` browser view, with only a few minor differences.

For general information on how to query the Plone catalog, please refer to the Plone Documentation on Querying.

Query options

In case you want to supply query options to a query against a particular index, you'll need to flatten the corresponding query dictionary and use a dotted notation to indicate nesting.

For example, to specify the `depth` query option for a path query, the original query as a Python dictionary would look like this:

```
query = {'path': {'query': '/folder',
                  'depth': 2}}
```

This dictionary will need to be flattened in dotted notation in order to pass it in a query string:

```
GET /plone/@search?path.query=%2Ffolder&path.depth=2
```

Again, this is very similar to how Record Arguments are parsed by ZPublisher, except that you can omit the `:record` suffix.

Data types in queries

Because HTTP query strings contain no information about data types, any query string parameter value ends up as a string in the Zope's request. This means, that for values types that aren't string, these data types need to be reconstructed on the server side in plone.restapi.

For most index types and their query values and query options, plone.restapi can handle this for you. If you pass it `path.query=foo&path.depth=1`, it has the necessary knowledge about the `ExtendedPathIndex`'s options to turn the string `1` for the `depth` argument back into an integer before passing the query on to the catalog.

However, certain index types (a `FieldIndex` for example) may take arbitrary data types as query values. In that case, `plone.restapi` simply can't know what data type to cast your query value to, and you'll need to specify it using ZPublisher type hints:

```
GET /plone/@search?numeric_field=42:int HTTP/1.1
Accept: application/json
```

Please refer to the Documentation on Argument Conversion in ZPublisher for details.

1.7.2 Retrieving additional metadata

By default the results are represented as summaries that only contain the most basic information about the items, like their URL and title. If you need to retrieve additional metadata columns, you can do so by specifying the additional column names in the `metadata_fields` parameter:

```
GET /plone/@search?SearchableText=lorem&metadata_fields=modified HTTP/1.1
Accept: application/json
```

The metadata from those columns then will be included in the results. In order to specify multiple columns, simply repeat the query string parameter once for every column name (the `requests` module will do this automatically for you if you pass it a list of values for a query string parameter).

In order to retrieve all metadata columns that the catalog knows about, use `metadata_fields=_all`.

1.8 Additional Endpoints (MOCKS ONLY)

These are mocked endpoints for use during the Barcelona 2016 Sprint.

Warning: These endpoints simply deliver mocked content (canned responses). They are not intended for actual use, but instead serve as a skeleton to flesh out different aspects of the API during the Barcelona 2016 sprint.

1.8.1 Theme

Requesting for overridden resources

```
GET /:portal/@theme HTTP/1.1
Host: localhost:8080
Accept: application/json
```

Example:

```
GET /plone/@theme?resource=%2Fstyle%2Fmain.css
Accept: application/json

HTTP 200 OK
content-type: application/json

{
  "href": "http://plone/++theme++mytheme/style/main.css"
}
```

1.8.2 Components

Get the required component(s)

```
GET /:portal/@components/:[id,] HTTP/1.1
Host: localhost:8080
Accept: application/json
```

Examples:

```
GET /plone/@components/navigation
Accept: application/json

HTTP 200 OK
content-type: application/json

[
  {
    "data": {
      "items": [
        {
          "title": "Home",
          "url": "http://localhost:55001/plone"
        },
        {
          "title": "Test Folder",
          "url": "http://localhost:55001/plone/robot-test-folder"
        },
        {
          "title": "Welcome to Plone",
          "url": "http://localhost:55001/plone/front-page"
        }
      ]
    },
    "id": "navigation"
  }
]
```

```
GET /plone/front-page/@components/breadcrumbs
Accept: application/json

HTTP 200 OK
content-type: application/json

[
  {
    "data": {
      "items": [
        {
          "title": "Welcome to Plone",
          "url": "http://localhost:55001/plone/front-page"
        }
      ]
    },
    "id": "breadcrumbs"
  }
]
```

1.8.3 Actions

Get the available actions for the given context

```
GET /:path/@actions HTTP/1.1
Host: localhost:8080
Accept: application/json
```

Example:

```
GET /plone/@actions
Accept: application/json

HTTP 200 OK
content-type: application/json

{
  "actions": [
    {
      "@id": "view",
      "category": "",
      "title": "View"
    },
    {
      "@id": "edit",
      "category": "",
      "title": "Edit"
    },
    {
      "@id": "Collection",
      "category": "factories",
      "title": "Collection"
    },
    {
      "@id": "Document",
      "category": "factories",
      "title": "Document"
    },
    {
      "@id": "reject",
      "category": "workflow",
      "title": "Send back"
    }
  ]
}
```

1.8.4 Framed Responses

```
GET /:path?frame=object HTTP/1.1
Host: localhost:8080
Accept: application/json
```

Example:

```
GET /plone/front-page?frame=object
Accept: application/json

HTTP 200 OK
content-type: application/json

{
  "@id": "http://localhost:55001/plone/front-page",
  "@type": "Document",
  "actions": {
    "@id": "http://localhost:55001/plone/front-page/@actions/"
  },
  "fields": {
    "UID": "1f699ffa110e45afb1ba502f75f7ec33",
    "allow_discussion": null,
    "changeNote": "",
    "contributors": [],
    "created": "2016-01-21T01:14:48+00:00",
    "creators": [
      "test_user_1_",
      "admin"
    ],
    "description": "Congratulations! You have successfully installed Plone.",
    "effective": null,
    "exclude_from_nav": false,
    "expires": null,
    "id": "front-page",
    "language": "",
    "modified": "2016-01-21T01:24:11+00:00",
    "parent": {
      "@id": "http://localhost:55001/plone",
      "@type": "Plone Site",
      "description": "",
      "title": "Plone site"
    },
    "relatedItems": [],
    "rights": "",
    "subjects": [],
    "table_of_contents": null,
    "text": {
      "content-type": "text/plain",
      "data": "If you're seeing this instead of the web site you were expecting, the owner of this we
      "encoding": "utf-8"
    },
    "title": "Welcome to Plone"
  },
  "template": "index",
  "versions": {
    "@id": "http://localhost:55001/plone/front-page/@versions/"
  },
  "workflow": {
    "@id": "http://localhost:55001/plone/front-page/@workflow/"
  }
}
```

1.8.5 Authentication

Login

```
POST /:path/@login HTTP/1.1
Host: localhost:8080
Accept: application/json
```

Example:

```
POST /plone/@login
Accept: application/json

{"username": "admin", "password": "admin"}

HTTP 200 OK
content-type: application/json

{
  "success": true,
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwiZnVsbG5hbWUiOiJGb28gYmFy
}
```

1.9 Customizing the API

1.9.1 Content serialization

Dexterity fields

The API automatically converts all field values to JSON compatible data, whenever possible. However, you might use fields which store data that cannot be automatically converted, or you might want to customize the representation of certain fields.

For extending or changing the serializing of certain dexterity fields you need to register an `IFieldSerializer`-adapter.

Example:

```
from plone.customfield.interfaces import ICustomField
from plone.dexterity.interfaces import IDexterityContent
from plone.restapi.interfaces import IFieldSerializer
from plone.restapi.serializer.converters import json_compatible
from plone.restapi.serializer.dxfields import DefaultFieldSerializer
from zope.component import adapter
from zope.interface import Interface
from zope.interface import implementer


@adapter(ICustomField, IDexterityContent, Interface)
@implementer(IFieldSerializer)
class CustomFieldSerializer(DefaultFieldSerializer):

    def __call__(self):
        value = self.get_value()
        if value is not None:
            # Do custom serializing here, e.g.:
            value = value.output()

        return json_compatible(value)
```

Register the adapter in ZCML:

```
<configure xmlns="http://namespaces.zope.org/zope">

    <adapter factory=".serializer.CustomFieldSerializer" />

</configure>
```

The `json_compatible` function recursively converts the value to JSON compatible data, when possible. When a value cannot be converted, a `TypeError` is raised. It is recommended to pass all values through `json_compatible` in order to validate and convert them.

For customizing a specific field instance, a named `IFieldSerializer` adapter can be registered. The name may either be the full dottedname of the field (`plone.app.dexterity.behaviors.exclfromnav.IExcludeFromNavigation.exclude_from_nav`) or the shortname of the field (`exclude_from_nav`).

CHAPTER 2

Introduction

plone.restapi is a RESTful hypermedia API for Plone.

2.1 RESTful Hypermedia API

REST stands for Representational State Transfer. It is a software architectural principle to create loosely coupled web APIs.

Most web APIs have a tight coupling between client and server. This makes them brittle and hard to change over time. It requires them not only to fully document every small detail of the API, but also write a client implementation that follows that specification 100% and breaks as soon as you change any detail.

A hypermedia API just provides an entry point to the API that contains hyperlinks the clients can follow. Just like a human user of a regular website, that knows the initial URL of a website and then follows hyperlinks to navigate through the site. This has the advantage that the client just needs to understand how to detect and follow links. The URL and other details of the API can change without breaking the client.

CHAPTER 3

Documentation

http://plonerestapi.readthedocs.org

CHAPTER 4

Roadmap

https://github.com/plone/plone.restapi/milestones

CHAPTER 5

Live Demo

Heroku live demo:: http://stormy-headland-44390.herokuapp.com/Plone/

Note: You will need some kind of API browser application to explore the API. We recommend to use Postman.

CHAPTER 6

Design Decisions

- A truly RESTful API (Hypermedia / HATEOAS / Linked-data)
- JSON is the main target formt, support other formats (HTML, XML) later
- Use HTTP headers (to set format and versioning, also provide URL-based option to make it easier for people to try it out)
- Versioning should be included (we will decide later, if we actually want to support multiple versions at the same time)
- Field names just map over (we will not try to clean up attributes or enforce naming standards like pep8 (e.g. isPrincipiaFoldish -> is_folderish)
- Dexterity only. We will not put effort into supporting Archetypes.

CHAPTER 7

Software Quality

- 100% Test Coverage
- 100% PEP8 compliant

CHAPTER 8

Further Reading

- REST in Practice: Hypermedia and Systems Architecture (Webber, Parastatidis, Robinson)

CHAPTER 9

Standards

- JSON-LD
- JSON Schema
- Schema.org
- Hydra
- Collection+JSON
- Siren

CHAPTER 10

License

The project is licensed under the GPLv2.

10.1 Appendix, Indices and tables

10.1.1 Types

Available content types in a Plone site can be listed and queried by accessing the `/@types` endpoint on portal root (requires an authenticated user):

Note: These docs are generated by code tests, therefore you will see some 'test' contenttypes appear here.

```
GET /plone/@types
Accept: application/json

HTTP 200 OK
content-type: application/json

[
  {
    "@id": "http://localhost:55001/plone/@types/Collection",
    "title": "Collection"
  },
  {
    "@id": "http://localhost:55001/plone/@types/Discussion Item",
    "title": "Discussion Item"
  },
  {
    "@id": "http://localhost:55001/plone/@types/DXTestDocument",
    "title": "DXTestDocument"
  },
  {
    "@id": "http://localhost:55001/plone/@types/Event",
    "title": "Event"
  },
  {
    "@id": "http://localhost:55001/plone/@types/File",
    "title": "File"
  },
  {
    "@id": "http://localhost:55001/plone/@types/Folder",
    "title": "Folder"
  },
  {
    "@id": "http://localhost:55001/plone/@types/Image",
    "title": "Image"
  },
  {
    "@id": "http://localhost:55001/plone/@types/Link",
    "title": "Link"
  },
  {
    "@id": "http://localhost:55001/plone/@types/News Item",
    "title": "News Item"
  },
  {
    "@id": "http://localhost:55001/plone/@types/Document",
    "title": "Document"
  },
  {
    "@id": "http://localhost:55001/plone/@types/ATTestDocument",
    "title": "ATTestDocument"
  },
  {
    "@id": "http://localhost:55001/plone/@types/ATTestFolder",
    "title": "ATTestFolder"
  }
]
```

To get the schema of a content type, access the `/@types` endpoint with the name of the content type, e.g.:

Note: Due to Plone's extensive language support, there are some long 'language' fields in here. Just scroll on...

```
GET /plone/@types/Document
Accept: application/json

HTTP 200 OK
content-type: application/json+schema

{
  "properties": {
    "allow_discussion": {
      "choices": [
        [
          "True",
          "Yes"
        ],
        [
          "False",
          "No"
        ]
      ],
      "description": "Allow discussion for this content object.",
      "enum": [
        "True",
        "False"
      ],
      "enumNames": [
        "Yes",
        "No"
      ],
      "title": "Allow discussion",
      "type": "string"
    },
    "changeNote": {
      "description": "Enter a comment that describes the changes you made.",
      "title": "Change Note",
      "type": "string"
    },
    "contributors": {
      "additionalItems": true,
      "description": "The names of people that have contributed to this item. Each contributor shoul
      "items": {
        "description": "",
        "title": "",
        "type": "string"
      },
      "title": "Contributors",
      "type": "array",
      "uniqueItems": true
    },
    "creators": {
      "additionalItems": true,
      "description": "Persons responsible for creating the content of this item. Please enter a list
      "items": {
        "description": "",
        "title": "",
        "type": "string"
      },
      "title": "Creators",
      "type": "array",
      "uniqueItems": true
    },
    "description": {
      "description": "Used in item listings and search results.",
      "minLength": 0,
      "title": "Summary",
      "type": "string"
    },
    "effective": {
      "description": "If this date is in the future, the content will not show up in listings and sea
      "title": "Publishing Date",
      "type": "string"
    },
    "exclude_from_nav": {
      "default": false,
      "description": "If selected, this item will not appear in the navigation tree",
      "title": "Exclude from navigation",
      "type": "boolean"
    },
    "expires": {
      "description": "When this date is reached, the content will nolonger be visible in listings and
      "title": "Expiration Date",
      "type": "string"
    },
    "language": {
      "choices": [
        [
          "af",
          "Afrikaans"
        ],
        [
          "ak",
          "Akan"
        ],
        [
          "ig",
          "As\u1ee5s\u1ee5 Igbo"
        ],
        [
          "ay",
          "Aymara"
        ],
        [
          "az",
          "Az\u0259ri T\u00fcrk\u00e7\u0259si"
        ],
        [
          "id",
          "Bahasa Indonesia"
        ],
        [
          "ms",
          "Bahasa Melayu"
        ],
        [
          "ba",
          "Bashkir"
        ],
        [
          "dz",
          "Bhutani"
        ],
        [
          "bh",
          "Bihari"
        ],
        [
          "bi",
          "Bislama"
        ],
        [
          "bs",
          "Bosanski"
        ],
        [
          "my",
          "Burmese"
        ],
        [
          "ca",
          "Catal\u00e0"
        ],
        [
          "ch",
          "Chamoru"
        ],
        [
          "co",
          "Corsu"
        ],
        [
          "cy",
          "Cymraeg"
        ],
        [
          "da",
          "Dansk"
        ],
        [
          "de",
          "Deutsch"
        ],
        [
          "nv",
          "Din\u00e9 bizaad"
        ],
        [
          "dv",
          "Divehi"
        ],
        [
          "et",
          "Eesti"
        ],
        [
          "en",
          "English"
        ],
        [
          "es",
          "Espa\u00f1ol"
        ],
        [
          "eo",
          "Esperanto"
        ],
        [
          "eu",
          "Euskara"
        ],
        [
          "ee",
          "E\u028begbe"
        ],
        [
          "fj",
          "Fiji"
        ],
        [
          "fr",
          "Fran\u00e7ais"
        ],
        [
          "fy",
          "Frysk"
        ],
        [
          "ff",
          "Fulfulde"
        ],
        [
          "fo",
          "F\u00f8royska"
        ],
        [
          "ga",
          "Gaeilge"
        ],
        [
          "gv",
          "Gaelg"
        ],
        [
          "gl",
          "Galego"
        ],
        [
          "kl",
          "Greenlandic"
        ],
        [
          "gn",
          "Guarani"
        ],
        [
          "gd",
          "G\u00e0idhlig"
        ],
        [
          "ki",
          "G\u0129k\u0169y\u0169"
        ],
        [
          "ho",
          "Hiri Motu"
        ],
        [
          "hr",
          "Hrvatski"
        ],
        [
          "io",
          "Ido"
        ],
        [
          "ia",
          "Interlingua"
        ],
        [
          "ie",
          "Interlingue"
        ],
        [
          "nr",
          "IsiNdebele"
        ],
        [
          "xh",
          "IsiXhosa"
        ],
        [
          "zu",
          "IsiZulu"
        ],
        [
          "it",
          "Italiano"
        ],
        [
          "ik",
          "I\u00f1upiaq"
        ],
        [
          "jv",
          "Javanese"
        ],
        [
          "mh",
          "Kajin M\u0327aje\u013c"
        ],
        [
          "kr",
          "Kanuri"
        ],
        [
          "kw",
          "Kernewek"
        ],
        [
          "kg",
          "KiKongo"
        ],
        [
          "rw",
          "Kinyarwanda"
        ],
        [
          "rn",
          "Kirundi"
        ],
        [
          "sw",
          "Kiswahili"
        ],
        [
        "ht",
        "Krey\u00f2l ayisyen"
      ],
      [
        "kj",
        "Kuanyama"
      ],
      [
        "ku",
        "Kurd\u00ed"
      ],
      [
        "la",
        "Latin"
      ],
      [
        "lv",
        "Latvie\u0161u"
      ],
      [
        "lt",
        "Lietuviskai"
      ],
      [
        "li",
        "Limburgs"
      ],
      [
        "ln",
        "Lingala"
      ],
      [
        "lg",
        "Luganda"
      ],
      [
        "lb",
        "L\u00ebtzebuergesch"
      ],
      [
        "hu",
        "Magyar"
      ],
      [
        "mg",
        "Malagasy"
      ],
      [
        "mt",
        "Malti"
      ],
      [
        "mi",
        "Maori"
      ],
      [
        "mo",
        "Moldavian"
        ],
        [
          "na",
          "Nauru"
        ],
        [
          "nd",
          "Ndebele (North)"
        ],
        [
          "nl",
          "Nederlands"
        ],
        [
          "no",
          "Norsk"
        ],
        [
          "nb",
          "Norsk bokm\u00e5l"
        ],
        [
          "se",
          "Northern S\u00e1mi"
        ],
        [
          "ii",
          "Nuosu"
        ],
        [
          "nn",
          "Nynorsk"
        ],
        [
          "oc",
          "Occitan"
        ],
        [
          "om",
          "Oromo"
        ],
        [
          "hz",
          "Otjiherero"
        ],
        [
          "vk",
          "Ovalingo"
        ],
        [
          "ng",
          "Owambo"
        ],
        [
          "pl",
          "Polski"
        ],
        [
          "pt",
          "Portugu\u00eas"
        ],
        [
          "qu",
          "Quechua"
        ],
        [
          "ty",
          "Reo Tahiti"
        ],
        [
          "rm",
          "Rhaeto-Romance"
        ],
        [
          "ro",
          "Rom\u00e2n\u0103"
        ],
        [
          "sm",
          "Samoan"
        ],
        [
          "sg",
          "Sangho"
        ],
        [
          "sh",
          "Serbo-Croatian"
        ],
        [
          "st",
          "Sesotho"
        ],
        [
          "tn",
          "Setswana"
        ],
        [
          "sn",
          "Shona"
        ],
        [
          "sq",
          "Shqip"
        ],
        [
          "ss",
          "SiSwati"
        ],
        [
          "sd",
          "Sindhi"
        ],
        [
          "si",
          "Singhalese"
        ],
        [
          "sk",
          "Sloven\u010dina"
        ],
        [
          "sl",
          "Sloven\u0161\u010dina"
        ],
        [
          "so",
          "Somali"
        ],
        [
          "su",
          "Sudanese"
        ],
        [
          "fi",
          "Suomi"
        ],
        [
          "sv",
          "Svenska"
        ],
        [
          "tl",
          "Tagalog"
        ],
        [
          "vi",
          "Ti\u1ebfng Vi\u1ec7t"
        ],
        [
          "to",
          "Tonga"
        ],
        [
          "lu",
          "Tshiluba"
        ],
        [
          "ve",
          "Tshiven\u1e13a"
        ],
        [
          "tw",
          "Twi"
        ],
        [
          "tr",
          "T\u00fcrk\u00e7e"
        ],
        [
          "ug",
          "Uigur"
        ],
        [
          "vo",
          "Volap\u00fck"
        ],
        [
          "wa",
          "Walon"
        ],
        [
          "wo",
          "Wolof"
        ],
        [
          "ts",
          "Xitsonga"
        ],
        [
          "yo",
          "Yor\u00f9b\u00e1"
        ],
        [
          "za",
          "Zhuang"
        ],
        [
          "an",
          "aragon\u00e9s"
        ],
        [
          "ae",
          "avesta"
        ],
        [
          "bm",
          "bamanankan"
        ],
        [
          "br",
          "brezhoneg"
        ],
        [
          "ny",
          "chiChe\u0175a"
        ],
        [
          "sc",
          "sardu"
        ],
        [
          "is",
          "\u00cdslenska"
        ],
        [
          "cs",
          "\u010ce\u0161tina"
        ],
        [
          "el",
          "\u0395\u03bb\u03bb\u03b7\u03bd\u03b9\u03ba\u03ac"
        ],
        [
          "uz",
          "\u040e\u0437\u0431\u0435\u043a\u0447\u0430"
        ],
        [
          "be",
          "\u0411\u0435\u043b\u0430\u0440\u0443\u0441\u043a\u0456"
        ],
        [
          "bg",
          "\u0411\u044a\u043b\u0433\u0430\u0440\u0441\u043a\u0438"
        ],
        [
          "ky",
          "\u041a\u044b\u0440\u0433\u044b\u0437"
        ],
        [
          "mk",
          "\u041c\u0430\u043a\u0435\u0434\u043e\u043d\u0441\u043a\u0438"
        ],
        [
          "mn",
          "\u041c\u043e\u043d\u0433\u043e\u043b"
        ],
        [
          "ru",
          "\u0420\u0443\u0441\u0441\u043a\u0438\u0439"
        ],
        [
          "tg",
          "\u0422\u043e\u04b7\u0438\u043a\u0438"
        ],
        [
          "uk",
          "\u0423\u043a\u0440\u0430\u0457\u043d\u0441\u044c\u043a\u0430"
        ],
        [
          "ab",
          "\u0431\u044b\u0437\u0448\u04d9\u0430"
        ],
        [
          "os",
          "\u0438\u0440\u043e\u043d \u00e6\u0432\u0437\u0430\u0433"
        ],
        [
          "kv",
          "\u043a\u043e\u043c\u0438 \u043a\u044b\u0432"
        ],
        [
          "aa",
          "\u043c\u0430\u0433I\u0430\u0440\u0443\u043b \u043c\u0430\u0446I"
        ],
        [
          "ce",
          "\u043d\u043e\u0445\u0447\u0438\u0439\u043d \u043c\u043e\u0442\u0442"
        ],
        [
          "sr",
          "\u0441\u0440\u043f\u0441\u043a\u0438"
        ],
        [
          "tt",
          "\u0442\u0430\u0442\u0430\u0440\u0447\u0430"
        ],
        [
          "tk",
          "\u0442\u04af\u0440\u043am\u0435\u043d\u0447\u0435"
        ],
        [
          "cv",
          "\u0447\u04d1\u0432\u0430\u0448 \u0447\u04d7\u043b\u0445\u0438"
        ],
        [
          "cu",
          "\u0469\u0437\u044b\u043a\u044a \u0441\u043b\u043e\u0432\u0463\u043d\u044c\u0441\u043a\u044
        ],
        [
          "hy",
          "\u0540\u0561\u0575\u0565\u0580\u0567\u0576"
        ],
        [
          "he",
          "\u05e2\u05d1\u05e8\u05d9\u05ea"
        ],
        [
          "yi",
          "\u05f2\u05b4\u05d3\u05d9\u05e9"
        ],
        [
          "ur",
          "\u0627\u0631\u062f\u0648"
        ],
        [
          "ar",
          "\u0627\u0644\u0639\u0631\u0628\u064a\u0629"
        ],
        [
          "fa",
          "\u0641\u0627\u0631\u0633\u06cc"
        ],
        [
          "ha",
          "\u0647\u064e\u0648\u064f\u0633"
        ],
        [
          "ps",
          "\u067e\u069a\u062a\u0648"
        ],
        [
          "ks",
          "\u0915\u093e\u093d\u0936\u0941\u0930"
        ],
        [
          "ne",
          "\u0928\u0947\u092a\u093e\u0932\u0940"
        ],
        [
          "pi",
          "\u092a\u093e\u0934\u093f"
        ],
        [
          "mr",
          "\u092e\u0930\u093e\u0920\u0940"
        ],
        [
          "sa",
          "\u0938\u0902\u0938\u094d\u0915\u0943\u0924"
        ],
        [
          "hi",
          "\u0939\u093f\u0902\u0926\u0940"
        ],
        [
          "as",
          "\u0985\u09b8\u09ae\u09bf\u09df\u09be"
        ],
        [
          "bn",
          "\u09ac\u09be\u0982\u09b2\u09be"
        ],
        [
          "pa",
          "\u0a2a\u0a70\u0a1c\u0a3e\u0a2c\u0a40"
        ],
        [
          "gu",
          "\u0a97\u0ac1\u0a9c\u0ab0\u0abe\u0aa4\u0ac0"
        ],
        [
          "or",
          "\u0b13\u0b5c\u0b3f\u0b06"
        ],
        [
          "ta",
          "\u0ba4\u0bae\u0bbf\u0bb4"
        ],
        [
          "te",
          "\u0c24\u0c46\u0c32\u0c41\u0c17\u0c41"
        ],
        [
          "kn",
          "\u0c95\u0ca8\u0ccd\u0ca8\u0ca1"
        ],
        [
          "ml",
          "\u0d2e\u0d32\u0d2f\u0d3e\u0d33\u0d02"
        ],
        [
          "th",
          "\u0e44\u0e17\u0e22"
        ],
        [
          "lo",
          "\u0e9e\u0eb2\u0eaa\u0eb2\u0ea5\u0eb2\u0ea7"
        ],
        [
          "bo",
          "\u0f56\u0f7c\u0f51\u0f0b\u0f66\u0f90\u0f51\u0f0b"
        ],
        [
          "ka",
          "\u10e5\u10d0\u10e0\u10d7\u10e3\u10da\u10d8"
        ],
        [
          "ti",
          "\u1275\u130d\u122d\u129b"
        ],
        [
          "am",
          "\u12a0\u121b\u122d\u129b"
        ],
        [
          "iu",
          "\u1403\u14c4\u1483\u144e\u1450\u1466"
        ],
        [
          "oj",
          "\u140a\u14c2\u1511\u14c8\u142f\u14a7\u140e\u14d0"
        ],
        [
          "cr",
          "\u14c0\u1426\u1403\u152d\u140d\u140f\u1423"
        ],
        [
          "km",
          "\u1781\u17d2\u1798\u17c2\u179a"
        ],
        [
          "zh",
          "\u4e2d\u6587"
        ],
        [
          "ja",
          "\u65e5\u672c\u8a9e"
        ],
        [
          "ko",
          "\ud55c\uad6d\uc5b4"
        ],
        [
          "kk",
          "\ufed7\ufe8e\ufeaf\ufe8d\ufed7\ufeb8\ufe8e"
        ]
      ],
      "default": "en",
      "description": "",
      "enum": [
        "af",
        "ak",
        "ig",
        "ay",
        "az",
        "id",
        "ms",
        "ba",
        "dz",
        "bh",
        "bi",
        "bs",
        "my",
        "ca",
        "ch",
        "co",
        "cy",
        "da",
        "de",
        "nv",
        "dv",
        "et",
        "en",
        "es",
        "eo",
        "eu",
        "ee",
        "fj",
        "fr",
        "fy",
        "ff",
        "fo",
        "ga",
        "gv",
        "gl",
        "kl",
        "gn",
        "gd",
        "ki",
        "ho",
        "hr",
        "io",
        "ia",
        "ie",
        "nr",
        "xh",
        "zu",
        "it",
        "ik",
        "jv",
        "mh",
        "kr",
        "kw",
        "kg",
        "rw",
        "rn",
        "sw",
        "ht",
        "kj",
        "ku",
        "la",
        "lv",
        "lt",
        "li",
        "ln",
        "lg",
        "lb",
        "hu",
        "mg",
        "mt",
        "mi",
        "mo",
        "na",
        "nd",
        "nl",
        "no",
        "nb",
        "se",
        "ii",
        "nn",
        "oc",
        "om",
        "hz",
        "vk",
        "ng",
        "pl",
        "pt",
        "qu",
        "ty",
        "rm",
        "ro",
        "sm",
        "sg",
        "sh",
        "st",
        "tn",
        "sn",
        "sq",
        "ss",
        "sd",
        "si",
        "sk",
        "sl",
        "so",
        "su",
        "fi",
        "sv",
        "tl",
        "vi",
        "to",
        "lu",
        "ve",
        "tw",
        "tr",
        "ug",
        "vo",
        "wa",
        "wo",
        "ts",
        "yo",
        "za",
        "an",
        "ae",
        "bm",
        "br",
        "ny",
        "sc",
        "is",
        "cs",
        "el",
        "uz",
        "be",
        "bg",
        "ky",
        "mk",
        "mn",
        "ru",
        "tg",
        "uk",
        "ab",
        "os",
        "kv",
        "aa",
        "ce",
        "sr",
        "tt",
        "tk",
        "cv",
        "cu",
        "hy",
        "he",
        "yi",
        "ur",
        "ar",
        "fa",
        "ha",
        "ps",
        "ks",
        "ne",
        "pi",
        "mr",
        "sa",
        "hi",
        "as",
        "bn",
        "pa",
        "gu",
        "or",
        "ta",
        "te",
        "kn",
        "ml",
        "th",
        "lo",
        "bo",
        "ka",
        "ti",
        "am",
        "iu",
        "oj",
        "cr",
        "km",
        "zh",
        "ja",
        "ko",
        "kk"
      ],
      "enumNames": [
        "Afrikaans",
        "Akan",
        "As\u1ee5s\u1ee5 Igbo",
        "Aymara",
        "Az\u0259ri T\u00fcrk\u00e7\u0259si",
        "Bahasa Indonesia",
        "Bahasa Melayu",
        "Bashkir",
        "Bhutani",
        "Bihari",
        "Bislama",
        "Bosanski",
        "Burmese",
        "Catal\u00e0",
        "Chamoru",
        "Corsu",
        "Cymraeg",
        "Dansk",
        "Deutsch",
        "Din\u00e9 bizaad",
        "Divehi",
        "Eesti",
        "English",
        "Espa\u00f1ol",
        "Esperanto",
        "Euskara",
        "E\u028begbe",
        "Fiji",
        "Fran\u00e7ais",
        "Frysk",
        "Fulfulde",
        "F\u00f8royska",
        "Gaeilge",
        "Gaelg",
        "Galego",
        "Greenlandic",
        "Guarani",
        "G\u00e0idhlig",
        "G\u0129k\u0169y\u0169",
        "Hiri Motu",
        "Hrvatski",
        "Ido",
        "Interlingua",
        "Interlingue",
        "IsiNdebele",
        "IsiXhosa",
        "IsiZulu",
        "Italiano",
        "I\u00f1upiaq",
        "Javanese",
        "Kajin M\u0327aje\u013c",
        "Kanuri",
        "Kernewek",
        "KiKongo",
        "Kinyarwanda",
        "Kirundi",
        "Kiswahili",
        "Krey\u00f2l ayisyen",
        "Kuanyama",
        "Kurd\u00ed",
        "Latin",
        "Latvie\u0161u",
        "Lietuviskai",
        "Limburgs",
        "Lingala",
        "Luganda",
        "L\u00ebtzebuergesch",
        "Magyar",
        "Malagasy",
        "Malti",
        "Maori",
        "Moldavian",
        "Nauru",
        "Ndebele (North)",
        "Nederlands",
        "Norsk",
        "Norsk bokm\u00e5l",
        "Northern S\u00e1mi",
        "Nuosu",
        "Nynorsk",
        "Occitan",
        "Oromo",
        "Otjiherero",
        "Ovalingo",
        "Owambo",
        "Polski",
        "Portugu\u00eas",
        "Quechua",
        "Reo Tahiti",
        "Rhaeto-Romance",
        "Rom\u00e2n\u0103",
        "Samoan",
        "Sangho",
        "Serbo-Croatian",
        "Sesotho",
        "Setswana",
        "Shona",
        "Shqip",
        "SiSwati",
        "Sindhi",
        "Singhalese",
        "Sloven\u010dina",
        "Sloven\u0161\u010dina",
        "Somali",
        "Sudanese",
        "Suomi",
        "Svenska",
        "Tagalog",
        "Ti\u1ebfng Vi\u1ec7t",
        "Tonga",
        "Tshiluba",
        "Tshiven\u1e13a",
        "Twi",
        "T\u00fcrk\u00e7e",
        "Uigur",
        "Volap\u00fck",
        "Walon",
        "Wolof",
        "Xitsonga",
        "Yor\u00f9b\u00e1",
        "Zhuang",
        "aragon\u00e9s",
        "avesta",
        "bamanankan",
        "brezhoneg",
        "chiChe\u0175a",
        "sardu",
        "\u00cdslenska",
        "\u010ce\u0161tina",
        "\u0395\u03bb\u03bb\u03b7\u03bd\u03b9\u03ba\u03ac",
        "\u040e\u0437\u0431\u0435\u043a\u0447\u0430",
        "\u0411\u0435\u043b\u0430\u0440\u0443\u0441\u043a\u0456",
        "\u0411\u044a\u043b\u0433\u0430\u0440\u0441\u043a\u0438",
        "\u041a\u044b\u0440\u0433\u044b\u0437",
        "\u041c\u0430\u043a\u0435\u0434\u043e\u043d\u0441\u043a\u0438",
        "\u041c\u043e\u043d\u0433\u043e\u043b",
        "\u0420\u0443\u0441\u0441\u043a\u0438\u0439",
        "\u0422\u043e\u04b7\u0438\u043a\u0438",
        "\u0423\u043a\u0440\u0430\u0457\u043d\u0441\u044c\u043a\u0430",
        "\u0431\u044b\u0437\u0448\u04d9\u0430",
        "\u0438\u0440\u043e\u043d \u00e6\u0432\u0437\u0430\u0433",
        "\u043a\u043e\u043c\u0438 \u043a\u044b\u0432",
        "\u043c\u0430\u0433I\u0430\u0440\u0443\u043b \u043c\u0430\u0446I",
        "\u043d\u043e\u0445\u0447\u0438\u0439\u043d \u043c\u043e\u0442\u0442",
        "\u0441\u0440\u043f\u0441\u043a\u0438",
        "\u0442\u0430\u0442\u0430\u0440\u0447\u0430",
        "\u0442\u04af\u0440\u043am\u0435\u043d\u0447\u0435",
        "\u0447\u04d1\u0432\u0430\u0448 \u0447\u04d7\u043b\u0445\u0438",
        "\u0469\u0437\u044b\u043a\u044a \u0441\u043b\u043e\u0432\u0463\u043d\u044c\u0441\u043a\u044a
        "\u0540\u0561\u0575\u0565\u0580\u0567\u0576",
        "\u05e2\u05d1\u05e8\u05d9\u05ea",
        "\u05f2\u05b4\u05d3\u05d9\u05e9",
        "\u0627\u0631\u062f\u0648",
        "\u0627\u0644\u0639\u0631\u0628\u064a\u0629",
        "\u0641\u0627\u0631\u0633\u06cc",
        "\u0647\u064e\u0648\u064f\u0633",
        "\u067e\u069a\u062a\u0648",
        "\u0915\u093e\u093d\u0936\u0941\u0930",
        "\u0928\u0947\u092a\u093e\u0932\u0940",
        "\u092a\u093e\u0934\u093f",
        "\u092e\u0930\u093e\u0920\u0940",
        "\u0938\u0902\u0938\u094d\u0915\u0943\u0924",
        "\u0939\u093f\u0902\u0926\u0940",
        "\u0985\u09b8\u09ae\u09bf\u09df\u09be",
        "\u09ac\u09be\u0982\u09b2\u09be",
        "\u0a2a\u0a70\u0a1c\u0a3e\u0a2c\u0a40",
        "\u0a97\u0ac1\u0a9c\u0ab0\u0abe\u0aa4\u0ac0",
        "\u0b13\u0b5c\u0b3f\u0b06",
        "\u0ba4\u0bae\u0bbf\u0bb4",
        "\u0c24\u0c46\u0c32\u0c41\u0c17\u0c41",
        "\u0c95\u0ca8\u0ccd\u0ca8\u0ca1",
        "\u0d2e\u0d32\u0d2f\u0d3e\u0d33\u0d02",
        "\u0e44\u0e17\u0e22",
        "\u0e9e\u0eb2\u0eaa\u0eb2\u0ea5\u0eb2\u0ea7",
        "\u0f56\u0f7c\u0f51\u0f0b\u0f66\u0f90\u0f51\u0f0b",
        "\u10e5\u10d0\u10e0\u10d7\u10e3\u10da\u10d8",
        "\u1275\u130d\u122d\u129b",
        "\u12a0\u121b\u122d\u129b",
        "\u1403\u14c4\u1483\u144e\u1450\u1466",
        "\u140a\u14c2\u1511\u14c8\u142f\u14a7\u140e\u14d0",
        "\u14c0\u1426\u1403\u152d\u140d\u140f\u1423",
        "\u1781\u17d2\u1798\u17c2\u179a",
        "\u4e2d\u6587",
        "\u65e5\u672c\u8a9e",
        "\ud55c\uad6d\uc5b4",
        "\ufed7\ufe8e\ufeaf\ufe8d\ufed7\ufeb8\ufe8e"
      ],
      "title": "Language",
      "type": "string"
    },
    "relatedItems": {
      "additionalItems": true,
      "default": [],
      "description": "",
      "items": {
        "description": "",
        "title": "Related",
        "type": "string"
      },
      "title": "Related Items",
      "type": "array",
      "uniqueItems": true
    },
    "rights": {
      "description": "Copyright statement or other rights information on this item.",
      "minLength": 0,
      "title": "Rights",
      "type": "string"
    },
    "subjects": {
      "additionalItems": true,
      "description": "Tags are commonly used for ad-hoc organization of content.",
      "items": {
        "description": "",
        "title": "",
        "type": "string"
      },
      "title": "Tags",
      "type": "array",
      "uniqueItems": true
    },
    "table_of_contents": {
      "description": "If selected, this will show a table of contents at the top of the page.",
      "title": "Table of contents",
      "type": "boolean"
    },
    "text": {
      "description": "",
      "properties": {
        "text.encoding": {
          "description": "Mainly used internally",
          "title": "Default encoding for the value",
          "type": "string"
        },
        "text.mimeType": {
          "description": "",
          "title": "MIME type",
          "type": "string"
        },
        "text.output": {
          "description": "May be None if the transform cannot be completed",
          "minLength": 0,
          "title": "Transformed value in the target MIME type",
          "type": "string"
        },
        "text.outputMimeType": {
          "description": "",
          "title": "Default output MIME type",
          "type": "string"
        },
        "text.raw": {
          "description": "",
          "minLength": 0,
          "title": "Raw value in the original MIME type",
          "type": "string"
        },
        "text.raw_encoded": {
          "description": "Mainly used internally",
          "title": "Get the raw value as an encoded string",
          "type": "string"
        }
      },
      "title": "Text",
      "type": "object"
    },
    "title": {
      "description": "",
      "title": "Title",
      "type": "string"
    }
  },
  "required": [
    "title",
    "exclude_from_nav"
  ],
  "title": "Page",
  "type": "object"
}
```

The content type schema uses the JSON Schema format.

10.1.2 HTTP Status Codes

This is the list of status codes that are used in plone.restapi. Here is a full list of all HTTP status codes.

200 OK Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request, the response will contain an entity describing or containing the result of the action.

201 Created The request has been fulfilled and resulted in a new resource being created.

204 No Content The server successfully processed the request, but is not returning any content. Usually used as a response to a successful delete request.

2xx Success This class of status codes indicates the action requested by the client was received, understood, accepted and processed successfully.

400 Bad Request The server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing)

401 Unauthorized Similar to 403 Forbidden, but specifically for use when authentication is required and has failed or has not yet been provided. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource.

403 Forbidden The request was a valid request, but the server is refusing to respond to it. Unlike a 401 Unauthorized response, authenticating will make no difference.

404 Not Found The requested resource could not be found but may be available again in the future. Subsequent requests by the client are permissible.

405 Method Not Allowed A request method is not supported for the requested resource; for example, a GET request on a form which requires data to be presented via POST, or a PUT request on a read-only resource.

409 Conflict Indicates that the request could not be processed because of conflict in the request, such as an edit conflict in the case of multiple updates.

4xx Client Error The 4xx class of status code is intended for cases in which the client seems to have errored.

500 Internal Server Error The server failed to fulfill an apparently valid request.

5xx Server Error The server failed to fulfill an apparently valid request.

10.1.3 Glossary

REST REST stands for Representational State Transfer. It is a software architectural principle to create loosely coupled web APIs.

workflow A concept in Plone (and other CMS's) whereby a content object can be in a number of states (private, public, etcetera) and uses transitions to change between them (e.g. "publish", "approve", "reject", "retract"). See the Plone docs on Workflow

- genindex