<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Authentication</td>
<td>29</td>
</tr>
<tr>
<td>3</td>
<td>Explore the API using Postman</td>
<td>33</td>
</tr>
<tr>
<td>4</td>
<td>Content Manipulation</td>
<td>37</td>
</tr>
<tr>
<td>5</td>
<td>Volto Blocks support</td>
<td>53</td>
</tr>
<tr>
<td>6</td>
<td>History</td>
<td>57</td>
</tr>
<tr>
<td>7</td>
<td>Batching</td>
<td>61</td>
</tr>
<tr>
<td>8</td>
<td>Add-ons</td>
<td>65</td>
</tr>
<tr>
<td>9</td>
<td>Comments</td>
<td>71</td>
</tr>
<tr>
<td>10</td>
<td>Copy / Move</td>
<td>77</td>
</tr>
<tr>
<td>11</td>
<td>Expansion</td>
<td>81</td>
</tr>
<tr>
<td>12</td>
<td>Portal Actions</td>
<td>91</td>
</tr>
<tr>
<td>13</td>
<td>Workflow</td>
<td>95</td>
</tr>
<tr>
<td>14</td>
<td>Working Copy</td>
<td>99</td>
</tr>
<tr>
<td>15</td>
<td>Locking</td>
<td>107</td>
</tr>
<tr>
<td>16</td>
<td>Sharing</td>
<td>113</td>
</tr>
<tr>
<td>17</td>
<td>Registry</td>
<td>119</td>
</tr>
<tr>
<td>18</td>
<td>Types</td>
<td>131</td>
</tr>
<tr>
<td>19</td>
<td>Types Schema</td>
<td>161</td>
</tr>
<tr>
<td>20</td>
<td>Users</td>
<td>171</td>
</tr>
</tbody>
</table>
CHAPTER 1

Introduction

API Browser Quick Guide

It can make your life easier if you use some kind of API browser application to explore the API when diving into this documentation.

- We recommend to use the free Postman browser plugin.
- For easy onboarding take a look at our Explore the API using Postman Quick-Guide.

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 initial 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 /plone HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i http://nohost/plone -H 'Accept: application/json' --user admin:secret
```

httpie

```
http http://nohost/plone Accept:application/json -a admin:secret
```

python-requests
This uses so-called ‘content negotiation’

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:

```
HTTP/1.1 200 OK 
Content-Type: application/json
{
  "@components": {
    "actions": {
      "@id": "http://localhost:55001/plone/@actions"
    },
    "breadcrumbs": {
      "@id": "http://localhost:55001/plone/@breadcrumbs"
    },
    "contextnavigation": {
      "@id": "http://localhost:55001/plone/@contextnavigation"
    },
    "navigation": {
      "@id": "http://localhost:55001/plone/@navigation"
    }
  },
  "@id": "http://localhost:55001/plone",
  "@type": "Plone Site",
  "blocks": {},
  "blocks_layout": {},
  "description": "",
  "id": "plone",
  "is_folderish": true,
  "items": [
```

(continues on next page)
@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.

items 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 loosely 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 a request and response for a document. Click on the buttons below to show the different syntaxes for the request.

```plaintext
GET /plone/front-page HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```bash
curl -i http://nohost/plone/front-page -H 'Accept: application/json' --user admin:secret
```

httpie

```bash
http http://nohost/plone/front-page Accept:application/json -a admin:secret
```

python-requests

```python
requests.get('http://nohost/plone/front-page', headers={'Accept': 'application/json'},
            auth=('admin', 'secret'))
```

HTTP/1.1 200 OK
Content-Type: application/json

```json
{
    "@components": {
        "actions": {
            "@id": "http://localhost:55001/plone/front-page/@actions"
        },
        "breadcrumbs": {
            "@id": "http://localhost:55001/plone/front-page/@breadcrumbs"
        },
        "contextnavigation": {
```
"@id": "http://localhost:5501/plone/front-page/@contextnavigation",

"navigation": {
    "@id": "http://localhost:5501/plone/front-page/@navigation"
},

"types": {
    "@id": "http://localhost:5501/plone/front-page/@types"
},

"workflow": {
    "@id": "http://localhost:5501/plone/front-page/@workflow"
}

"@id": "http://localhost:5501/plone/front-page",

"@type": "Document",

"UID": "SomeUUID00000000000000000000000000000001",

"allow_discussion": false,

"changeNote": "",

"contributors": [],

"created": "1995-07-31T13:45:00",

"creators": [
    "test_user_1"
],

"description": "Congratulations! You have successfully installed Plone.",

"effective": null,

"exclude_from_nav": false,

"expires": null,

"id": "front-page",

"is_folderish": false,

"language": "",

"layout": "document_view",

"modified": "1995-07-31T17:30:00",

"next_item": {},

"parent": {
    "@id": "http://localhost:5501/plone",
    "@type": "Plone Site",
    "description": "",
    "title": "Plone site"
},

"previous_item": {},

"relatedItems": [],

"review_state": "private",

"rights": "",

"subjects": [],

"table_of_contents": null,

"text": {
    "content-type": "text/plain",
    "data": "<p>If you’re seeing this instead of the web site you were expecting, the owner of this web site has just installed Plone. Do not contact the Plone Team or the Plone mailing lists about this.</p>",
    "encoding": "utf-8"
},

"title": "Welcome to Plone",

"version": "current",

"versioning_enabled": true,

"working_copy": null,

"working_copy_of": null
And so on, see

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.

**Note:** For folderish types, collections or search results, the results will be **batched** if the size of the resultset exceeds the batch size. See *Batching* for more details on how to work with batched results.

1.2.1 Plone Portal Root:

http

```
GET /plone HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i http://nohost/plone -H 'Accept: application/json' --user admin:secret
```

httpie

```
http http://nohost/plone Accept:application/json -a admin:secret
```

python-requests

```
requests.get('http://nohost/plone', headers={'Accept': 'application/json'},
             auth=('admin', 'secret'))
```

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "@components": {
    "actions": {
      "@id": "http://localhost:55001/plone/@actions"
    },
    "breadcrumbs": {
      "@id": "http://localhost:55001/plone/@breadcrumbs"
    },
    "contextnavigation": {
      "@id": "http://localhost:55001/plone/@contextnavigation"
    },
    "navigation": {
      "@id": "http://localhost:55001/plone/@navigation"
    }
  },
  "@id": "http://localhost:55001/plone",
  "@type": "Plone Site",
  "blocks": {}
}
```

(continues on next page)
"blocks_layout": {},
"description": "",
"id": "plone",
"is_folderish": true,
"items": [
  {
    "@id": "http://localhost:55001/plone/front-page",
    "@type": "Document",
    "description": "Congratulations! You have successfully installed Plone."
  },
  {
    "@id": "http://localhost:55001/plone/folder",
    "@type": "Document",
    "description": "Welcome to Plone"
  }
],
"items_total": 1,
"parent": {},
"title": "Plone site"

1.2.2 Plone Folder:

http

GET /plone/folder HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0

curl

curl -i http://nohost/plone/folder -H 'Accept: application/json' --user admin:secret

httpie


requests

requests.get('http://nohost/plone/folder', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
}]

"navigation": {
  "@id": "http://localhost:55001/plone/folder/@navigation"
},

"types": {
  "@id": "http://localhost:55001/plone/folder/@types"
},

"workflow": {
  "@id": "http://localhost:55001/plone/folder/@workflow"
}
},

"@id": "http://localhost:55001/plone/folder",

"@type": "Folder",

"UID": "SomeUUID000000000000000000000000002",

"allow_discussion": false,

"contributors": [],

"created": "1995-07-31T13:45:00",

"creators": [
  "test_user_1_
],

"description": "This is a folder with two documents",

"effective": null,

"exclude_from_nav": false,

"expires": null,

"id": "folder",

"is_folderish": true,

"items": [
  {
    "@id": "http://localhost:55001/plone/folder/doc1",
    "@type": "Document",
    "description": "",
    "review_state": "private",
    "title": "A document within a folder"
  },
  {
    "@id": "http://localhost:55001/plone/folder/doc2",
    "@type": "Document",
    "description": "",
    "review_state": "private",
    "title": "A document within a folder"
  }
],

"items_total": 2,

"language": "",

"layout": "listing_view",

"modified": "1995-07-31T17:30:00",

"nextPreviousEnabled": false,

"next_item": {},

"parent": {
  "@id": "http://localhost:55001/plone",
  "@type": "Plone Site",
  "description": "",
  "title": "Plone site"
},

"previous_item": {
  "@id": "http://localhost:55001/plone/front-page",
  "@type": "Document",
  "description": "",
  "review_state": "private",
  "title": "A document within a folder"
}
“description”: "Congratulations! You have successfully installed Plone.",
“title”: "Welcome to Plone"
},
“relatedItems”: [],
“review_state”: "private",
“rights”: "",
“subjects”: [],
“title”: "My Folder",
“version”: "current",
“working_copy”: null,
“working_copy_of”: null
}

1.2.3 Plone Document:

http

GET /plone/front-page HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0

curl

curl -i http://nohost/plone/front-page -H 'Accept: application/json' --user admin:secret

httpie

http http://nohost/plone/front-page Accept:application/json -a admin:secret

python-requests

requests.get('http://nohost/plone/front-page', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))

HTTP/1.1 200 OK
Content-Type: application/json

{  
  "@components": {  
    "actions": {  
      "@id": "http://localhost:55001/plone/front-page/@actions"
    },
    "breadcrumbs": {  
      "@id": "http://localhost:55001/plone/front-page/@breadcrumbs"
    },
    "contextnavigation": {  
      "@id": "http://localhost:55001/plone/front-page/@contextnavigation"
    },
    "navigation": {  
      "@id": "http://localhost:55001/plone/front-page/@navigation"
    },
    "types": {  
      "@id": "http://localhost:55001/plone/front-page/@types"
    }
  }
}
"workflow": {
    "@id": "http://localhost:55001/plone/front-page/@workflow"
}
,
"@id": "http://localhost:55001/plone/front-page",
"@type": "Document",
"UID": "SomeUUID0000000000000000000000001",
"allow_discussion": false,
"changeNote": "",
"contributors": [],
"created": "1995-07-31T13:45:00",
"creators": [
    "test_user_1_"
],
"description": "Congratulations! You have successfully installed Plone.",
"effective": null,
"exclude_from_nav": false,
"expires": null,
"id": "front-page",
"is_folderish": false,
"language": "",
"layout": "document_view",
"modified": "1995-07-31T17:30:00",
"next_item": {},
"parent": {
    "@id": "http://localhost:55001/plone",
    "@type": "Plone Site",
    "description": "",
    "title": "Plone site"
},
"previous_item": {},
"relatedItems": [],
"review_state": "private",
"rights": "",
"subjects": [],
"table_of_contents": null,
"text": {
    "content-type": "text/plain",
    "data": "<p>If you're seeing this instead of the web site you were expecting, the owner of this web site has just installed Plone. Do not contact the Plone Team or the Plone mailing lists about this.</p>",
    "encoding": "utf-8"
},
"title": "Welcome to Plone",
"version": "current",
"versioning_enabled": true,
"working_copy": null,
"working_copy_of": null
}

1.2.4 News Item:
Note: Here we show uuid1 as an example uid for all image scales because this documentation is autogenerated by the tests. When running in a real application, these uuid1 values will be exchanged by proper uuid4 values.

http

```plaintext
GET /plone/newsitem HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```plaintext
curl -i http://nohost/plone/newsitem -H 'Accept: application/json' --user admin:secret
```

httpie

```plaintext
http http://nohost/plone/newsitem Accept:application/json -a admin:secret
```

python-requests

```python
requests.get('http://nohost/plone/newsitem', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

```json
HTTP/1.1 200 OK
Content-Type: application/json

{
   "@components": {
      "actions": {
         "@id": "http://localhost:55001/plone/newsitem/@actions"
      },
      "breadcrumbs": {
         "@id": "http://localhost:55001/plone/newsitem/@breadcrumbs"
      },
      "contextnavigation": {
         "@id": "http://localhost:55001/plone/newsitem/@contextnavigation"
      },
      "navigation": {
         "@id": "http://localhost:55001/plone/newsitem/@navigation"
      },
      "types": {
         "@id": "http://localhost:55001/plone/newsitem/@types"
      },
      "workflow": {
         "@id": "http://localhost:55001/plone/newsitem/@workflow"
      }
   },
   "@id": "http://localhost:55001/plone/newsitem",
   "@type": "News Item",
   "UID": "SomeUUID000000000000000000000000000000002",
   "allow_discussion": false,
   "changeNote": "",
   "contributors": [],
   "created": "1995-07-31T13:45:00",
   "creators": [
```
"test_user_1_"
],
"description": "This is a news item",
"effective": null,
"exclude_from_nav": false,
"expires": null,
"id": "newsitem",
"image": {
  "content-type": "image/png",
  "download": "http://localhost:55001/plone/newsitem/@images/uuid1.png",
  "filename": "image.png",
  "height": 56,
  "scales": {
    "icon": {
      "download": "http://localhost:55001/plone/newsitem/@images/uuid1.png",
      "height": 8,
      "width": 32
    },
    "large": {
      "download": "http://localhost:55001/plone/newsitem/@images/uuid1.png",
      "height": 56,
      "width": 215
    },
    "listing": {
      "download": "http://localhost:55001/plone/newsitem/@images/uuid1.png",
      "height": 4,
      "width": 16
    },
    "mini": {
      "download": "http://localhost:55001/plone/newsitem/@images/uuid1.png",
      "height": 52,
      "width": 200
    },
    "preview": {
      "download": "http://localhost:55001/plone/newsitem/@images/uuid1.png",
      "height": 56,
      "width": 215
    },
    "thumb": {
      "download": "http://localhost:55001/plone/newsitem/@images/uuid1.png",
      "height": 33,
      "width": 128
    },
    "tile": {
      "download": "http://localhost:55001/plone/newsitem/@images/uuid1.png",
      "height": 16,
      "width": 64
    }
  }
},
"size": 1185,
1.2.5 Event:

http

GET /plone/event HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0

curl

curl -i http://nohost/plone/event -H 'Accept: application/json' --user admin:secret

httpie

http http://nohost/plone/event Accept:application/json -a admin:secret

python-requests
requests.get('http://nohost/plone/event', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))

HTTP/1.1 200 OK
Content-Type: application/json

{
    "@components": {
        "actions": {
            "@id": "http://localhost:55001/plone/event/@actions"
        },
        "breadcrumbs": {
            "@id": "http://localhost:55001/plone/event/@breadcrumbs"
        },
        "contextnavigation": {
            "@id": "http://localhost:55001/plone/event/@contextnavigation"
        },
        "navigation": {
            "@id": "http://localhost:55001/plone/event/@navigation"
        },
        "types": {
            "@id": "http://localhost:55001/plone/event/@types"
        },
        "workflow": {
            "@id": "http://localhost:55001/plone/event/@workflow"
        }
    },
    "@id": "http://localhost:55001/plone/event",
    "@type": "Event",
    "UID": "SomeUUID000000000000000000000000002",
    "allow_discussion": false,
    "attendees": [],
    "changeNote": "",
    "contact_email": null,
    "contact_name": null,
    "contact_phone": null,
    "contributors": [],
    "created": "1995-07-31T13:45:00",
    "creators": [
        "test_user_1_"
    ],
    "description": "This is an event",
    "effective": null,
    "end": "2013-01-01T12:00:00",
    "event_url": null,
    "exclude_from_nav": false,
    "expires": null,
    "id": "event",
    "is_folderish": false,
    "language": "",
    "layout": "event_view",
    "location": null,
    "modified": "1995-07-31T17:45:00",
    "next_item": {},
    "open_end": false,
    "parent": {
        "@id": "http://localhost:55001/plone",
        "@type": "Event"
    }
}
1.2.6 Image:

Note: Here we show uuid1 as an example uid for all image scales because this documentation is autogenerated by the tests. When running in a real application, these uuid1 values will be exchanged by proper uuid4 values.

http

```
GET /plone/image HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i http://nohost/plone/image -H 'Accept: application/json' --user admin:secret
```

httpie

```
http http://nohost/plone/image Accept:application/json -a admin:secret
```

python-requests

```
requests.get('http://nohost/plone/image', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "@components": {
        "actions": {
            "@id": "http://localhost:55001/plone/image/@actions"
        },
        "breadcrumbs": {
            "@id": "http://localhost:55001/plone/image/@breadcrumbs"
        },
        "contextnavigation": {
            "@id": "http://localhost:55001/plone/image/@contextnavigation"
        },
        "navigation": {
            "@id": "http://localhost:55001/plone/image/@navigation"
        },
        "types": {
            "@id": "http://localhost:55001/plone/image/@types"
        },
        " workflow": {
            "@id": "http://localhost:55001/plone/image/@workflow"
        }
    },
    "@id": "http://localhost:55001/plone/image",
    "@type": "Image",
    "UID": "SomeUUID00000000000000000000000000000002",
    "allow.discussion": false,
    "contributors": [],
    "created": "1995-07-31T13:45:00",
    "creators": [
        "test_user_1_"
    ],
    "description": "This is an image",
    "effective": null,
    "exclude_from_nav": false,
    "expires": null,
    "id": "image",
    "image": {
        "content-type": "image/png",
        "download": "http://localhost:55001/plone/image/@images/uuid1.png",
        "filename": "image.png",
        "height": 56,
        "scales": {
            "icon": {
                "download": "http://localhost:55001/plone/image/@images/uuid1.png",
                "height": 8,
                "width": 32
            },
            "large": {
                "download": "http://localhost:55001/plone/image/@images/uuid1.png",
                "height": 56,
                "width": 215
            },
            "listing": {
                "download": "http://localhost:55001/plone/image/@images/uuid1.png",
                "height": 4,
                "width": 32
            }
        }
    }
}

(continues on next page)
1.2.7 File:

http
GET /plone/file HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0

curl

curl -i http://nohost/plone/file -H 'Accept: application/json' --user 
   →admin:secret

httpie


python-requests

requests.get('http://nohost/plone/file', headers={'Accept': 'application/json
   →'}, auth=('admin', 'secret'))

HTTP/1.1 200 OK
Content-Type: application/json

{
   "@components": {
   "actions": {
       "@id": "http://localhost:55001/plone/file/@actions"
   },
   "breadcrumbs": {
       "@id": "http://localhost:55001/plone/file/@breadcrumbs"
   },
   "contextnavigation": {
       "@id": "http://localhost:55001/plone/file/@contextnavigation"
   },
   "navigation": {
       "@id": "http://localhost:55001/plone/file/@navigation"
   },
   "types": {
       "@id": "http://localhost:55001/plone/file/@types"
   },
   "workflow": {
       "@id": "http://localhost:55001/plone/file/@workflow"
   }
},
   "@id": "http://localhost:55001/plone/file",
   "@type": "File",
   "UID": "SomeUUID0000000000000000000000000002",
   "allow_discussion": false,
   "contributors": [],
   "created": "1995-07-31T13:45:00",
   "creators": [
       "test_user_1_"
   ],
   "description": "This is a file",
   "effective": null,
   "exclude_from_nav": false,
   "expires": null,
   "file": {
       "content-type": "application/pdf",
   }
   (continues on next page)
"download": "http://localhost:55001/plone/file/@download/file",
"filename": "file.pdf",
"size": 74429
],
"id": "file",
"is_folderish": false,
"language": "",
"layout": "file_view",
"modified": "1995-07-31T17:30:00",
"next_item": {},
"parent": {
"@id": "http://localhost:55001/plone",
"@type": "Plone Site",
"description": "",
"title": "Plone site"
},
"previous_item": {
"@id": "http://localhost:55001/plone/front-page",
"@type": "Document",
"description": "Congratulations! You have successfully installed Plone.",
"title": "Welcome to Plone"
},
"relatedItems": [],
"review_state": null,
"rights": "",
"subjects": [],
"title": "My File",
"version": "current",
"working_copy": null,
"working_copy_of": null
}

1.2.8 Link:

http

GET /plone/link HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0

curl

curl -i http://nohost/plone/link -H 'Accept: application/json' --user admin:secret

httpie

http http://nohost/plone/link Accept:application/json -a admin:secret

python-requests

requests.get('http://nohost/plone/link', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
HTTP/1.1 200 OK
Content-Type: application/json

{
  "@components": {
    "actions": {
      "@id": "http://localhost:55001/plone/link/@actions"
    },
    "breadcrumbs": {
      "@id": "http://localhost:55001/plone/link/@breadcrumbs"
    },
    "contextnavigation": {
      "@id": "http://localhost:55001/plone/link/@contextnavigation"
    },
    "navigation": {
      "@id": "http://localhost:55001/plone/link/@navigation"
    },
    "types": {
      "@id": "http://localhost:55001/plone/link/@types"
    },
    "workflow": {
      "@id": "http://localhost:55001/plone/link/@workflow"
    }
  },
  "@id": "http://localhost:55001/plone/link",
  "@type": "Link",
  "UID": "SomeUUID00000000000000000000000002",
  "allow_discussion": false,
  "changeNote": 
  "contributors": [],
  "created": "1995-07-31T13:45:00",
  "creators": [
    "test_user_1"
  ],
  "description": "This is a link",
  "effective": null,
  "exclude_from_nav": false,
  "expires": null,
  "id": "link",
  "is_folderish": false,
  "language": 
  "layout": "link_redirect_view",
  "modified": "1995-07-31T17:30:00",
  "next_item": {},
  "parent": {
    "@id": "http://localhost:55001/plone",
    "@type": "Plone Site",
    "description": 
    "title": "Plone site"
  },
  "previous_item": {
    "@id": "http://localhost:55001/plone/front-page",
    "@type": "Document",
    "description": "Congratulations! You have successfully installed Plone.",
    "title": "Welcome to Plone"
  },
  "remoteUrl": "http://localhost:55001/plone",
}
"review_state": "private",
"rights": "",
"subjects": [],
"title": "My Link",
"version": "current",
"versioning_enabled": true,
"working_copy": null,
"working_copy_of": null
}

1.2.9 Collection:

http

GET /plone/collection HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0

curl


httpie


python-requests

requests.get('http://nohost/plone/collection', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))

HTTP/1.1 200 OK
Content-Type: application/json

{
  "@components": {
    "actions": {
      "@id": "http://localhost:55001/plone/collection/@actions"
    },
    "breadcrumbs": {
      "@id": "http://localhost:55001/plone/collection/@breadcrumbs"
    },
    "contextnavigation": {
      "@id": "http://localhost:55001/plone/collection/@contextnavigation"
    },
    "navigation": {
      "@id": "http://localhost:55001/plone/collection/@navigation"
    },
    "types": {
      "@id": "http://localhost:55001/plone/collection/@types"
    },
    "workflow": {
      "@id": "http://localhost:55001/plone/collection/@workflow"
    }
  }
}
"@id": "http://localhost:55001/plone/collection",
"@type": "Collection",
"UUID": "SomeUUID000000000000000000000002",
"allow_discussion": false,
"contributors": [],
"created": "1995-07-31T13:45:00",
"creators": [
  "test_user_1",
],
"customViewFields": [
  {
    "title": "Title",
    "token": "Title"
  },
  {
    "title": "Creator",
    "token": "Creator"
  },
  {
    "title": "Type",
    "token": "Type"
  },
  {
    "title": "Last modified",
    "token": "ModificationDate"
  }
],
"description": "This is a collection with two documents",
"effective": null,
"exclude_from_nav": false,
"expires": null,
"id": "collection",
"is_folderish": false,
"item_count": 30,
"items": [
  {
    "@id": "http://localhost:55001/plone/front-page",
    "@type": "Document",
    "description": "Congratulations! You have successfully installed Plone.
    →",
    "review_state": "private",
    "title": "Welcome to Plone"
  },
  {
    "@id": "http://localhost:55001/plone/doc1",
    "@type": "Document",
    "description": "",
    "review_state": "private",
    "title": "Document 1"
  },
  {
    "@id": "http://localhost:55001/plone/doc2",
    "@type": "Document",
    "description": "",
    "review_state": "private",
    "title": "Document 2"
  },
You can also get all the data of each of the items of a collection, appending the `?fullobjects` parameter to the query: http

```
GET /plone/collection?fullobjects HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

```
```
httpie

```
```

Python-requests

```
requests.get('http://nohost/plone/collection?fullobjects', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

HTTP/1.1 200 OK
Content-Type: application/json

```
{
  "@components": {
    "actions": {
      "@id": "http://localhost:55001/plone/collection/@actions"
    },
    "breadcrumbs": {
      "@id": "http://localhost:55001/plone/collection/@breadcrumbs"
    },
    "contextnavigation": {
      "@id": "http://localhost:55001/plone/collection/@contextnavigation"
    },
    "navigation": {
      "@id": "http://localhost:55001/plone/collection/@navigation"
    },
    "types": {
      "@id": "http://localhost:55001/plone/collection/@types"
    },
    "workflow": {
      "@id": "http://localhost:55001/plone/collection/@workflow"
    }
  },
  "@id": "http://localhost:55001/plone/collection",
  "@type": "Collection",
  "UID": "SomeUUID000000000000000000000002",
  "allow_discussion": false,
  "contributors": [],
  "created": "1995-07-31T13:45:00",
  "creators": [
    "test_user_1_
  ],
  "customViewFields": [
    {
      "title": "Title",
      "token": "Title"
    },
    {
      "title": "Creator",
      "token": "Creator"
    },
    {
      "title": "Type",
      "token": "Type"
    },
    {
      "title": "Last modified",
      "token": "Last modified"
    }
  ]
}
```

(continues on next page)
"token": "ModificationDate"
}
],
"description": "This is a collection with two documents",
"effective": null,
"exclude_from_nav": false,
"expires": null,
"id": "collection",
"is_folderish": false,
"item_count": 30,
"items": [
  {
    "@components": {
      "actions": {
        "@id": "http://localhost:55001/plone/front-page/@actions"
      },
      "breadcrumbs": {
        "@id": "http://localhost:55001/plone/front-page/@breadcrumbs"
      },
      "contextnavigation": {
        "@id": "http://localhost:55001/plone/front-page/@contextnavigation"
      },
      "navigation": {
        "@id": "http://localhost:55001/plone/front-page/@navigation"
      },
      "types": {
        "@id": "http://localhost:55001/plone/front-page/@types"
      },
      "workflow": {
        "@id": "http://localhost:55001/plone/front-page/@workflow"
      }
    },
    "@id": "http://localhost:55001/plone/front-page",
    "@type": "Document",
    "UID": "SomeUUID000000000000000000000001",
    "allow_discussion": false,
    "changeNote": 
    "",
    "contributors": [],
    "created": "1995-07-31T13:45:00",
    "creators": [
      "test_user_1_"
    ],
    "description": "Congratulations! You have successfully installed Plone.
    "",
    "effective": null,
    "exclude_from_nav": false,
    "expires": null,
    "id": "front-page",
    "is_folderish": false,
    "language": 
    "",
    "layout": "document_view",
    "modified": "1995-07-31T17:30:00",
    "next_item": {
      "@id": "http://localhost:55001/plone/collection",
      "@type": "Collection",
      "description": "This is a collection with two documents",
      "title": "My Collection"
    }
  }
]
If you’re seeing this instead of the web site you were expecting, the owner of this web site has just installed Plone. Do not contact the Plone Team or the Plone mailing lists about this.

---

Welcome to Plone

---

1.2. Plone Content

---

25
"description": "",
"effective": null,
"exclude_from_nav": false,
"expires": null,
"id": "doc1",
"is_folderish": false,
"language": "",
"layout": "document_view",
"modified": "1995-07-31T17:30:00",

"next_item": {
  "@id": "http://localhost:55001/plone/doc2",
  "@type": "Document",
  "description": "",
  "title": "Document 2"
},

"parent": {
  "@id": "http://localhost:55001/plone",
  "@type": "Plone Site",
  "description": "",
  "title": "Plone site"
},

"previous_item": {
  "@id": "http://localhost:55001/plone/collection",
  "@type": "Collection",
  "description": "This is a collection with two documents",
  "title": "My Collection"
},

"relatedItems": [],
"review_state": "private",
"rights": "",
"subjects": [],
"table_of_contents": null,
"text": null,
"title": "Document 1",
"version": "current",
"versioning_enabled": true,
"working_copy": null,
"working_copy_of": null
},

"actions": {
  "@id": "http://localhost:55001/plone/doc2/@actions"
},

"breadcrumbs": {
  "@id": "http://localhost:55001/plone/doc2/@breadcrumbs"
},

"contextnavigation": {
  "@id": "http://localhost:55001/plone/doc2/@contextnavigation"
},

"navigation": {
  "@id": "http://localhost:55001/plone/doc2/@navigation"
},

"types": {
  "@id": "http://localhost:55001/plone/doc2/@types"
},

"workflow": {
```json
{
   "@id": "http://localhost:55001/plone/doc2",
   "@type": "Document",
   "UID": "SomeUUID000000000000000000000004",
   "allow_discussion": false,
   "changeNote": "",
   "contributors": [],
   "created": "1995-07-31T13:45:00",
   "creators": [
      "test_user_1_"
   ],
   "description": "",
   "effective": null,
   "exclude_from_nav": false,
   "expires": null,
   "id": "doc2",
   "is_folderish": false,
   "language": "",
   "layout": "document_view",
   "modified": "1995-07-31T17:30:00",
   "next_item": {},
   "parent": {
      "@id": "http://localhost:55001/plone",
      "@type": "Plone Site",
      "description": "",
      "title": "Plone site"
   },
   "previous_item": {
      "@id": "http://localhost:55001/plone/doc1",
      "@type": "Document",
      "description": "",
      "title": "Document 1"
   },
   "relatedItems": [],
   "review_state": "private",
   "rights": "",
   "subjects": [],
   "table_of_contents": null,
   "text": null,
   "title": "Document 2",
   "version": "current",
   "versioning_enabled": true,
   "working_copy": null,
   "working_copy_of": null
}
```

(continues on next page)
"title": "Document 1"
},
"parent": {
 "@id": "http://localhost:55001/plone",
 "@type": "Plone Site",
 "description": "",
 "title": "Plone site"
},
"previous_item": {
 "@id": "http://localhost:55001/plone/front-page",
 "@type": "Document",
 "description": "Congratulations! You have successfully installed Plone.",
 "title": "Welcome to Plone"
},
"query": [
 {
 "i": "portal_type",
 "o": "plone.app.querystring.operation.string.is",
 "v": "Document"
 }]
,"relatedItems": [],
"review_state": "private",
"rights": "",
"sort_on": null,
"sort_reversed": null,
"subjects": [],
"text": null,
"title": "My Collection",
"version": "current",
"working_copy": null,
"working_copy_of": null
}
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:

```
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
```

### 2.1 JSON Web Tokens (JWT)

plone.restapi includes a Plone PAS plugin for authentication with JWT. The plugin is installed automatically when installing the product.
2.1.1 Acquiring a token (@login)

A JWT token can be acquired by posting a user’s credentials to the @login endpoint. http

```plaintext
POST /plone/@login HTTP/1.1
Accept: application/json
Content-Type: application/json

{
    "login": "admin",
    "password": "secret"
}
```

curl

```plaintext
```

httpie

```plaintext
echo '{
    "login": "admin",
    "password": "secret"
}' | http POST http://nohost/plone/@login Accept:application/json Content-Type:application/json
```

python-requests

```python
requests.post('http://nohost/plone/@login', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'login': 'admin', 'password': 'secret'})
```

The server responds with a JSON object containing the token.

```plaintext
HTTP/1.1 200 OK
Content-Type: application/json

{
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
    eyJzdWIiOiJhZGljbIsImZ1bGxuYW1lIjoiIn0.krI8ep0NQHyYYObtX1lygI4NM2UbHUzqF7kKnx9JFo"
}
```

2.1.2 Authenticating with a token

The token can now be used in subsequent requests by including it in the Authorization header with the Bearer scheme: http

```plaintext
GET /plone/ HTTP/1.1
Accept: application/json
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
    eyJzdWIiOiJhZGljbIsImZ1bGxuYW1lIjoiIn0.krI8ep0NQHyYYObtX1lygI4NM2UbHUzqF7kKnx9JFo
```

curl

```plaintext
curl -i http://nohost/plone/ -H 'Accept: application/json' -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZGljbIsImZ1bGxuYW1lIjoiIn0.krI8ep0NQHyYYObtX1lygI4NM2UbHUzqF7kKnx9JFo'
```
2.1.3 Renewing a token (@login-renew)

By default the token will expire after 12 hours and thus must be renewed before expiration. To renew the token simply post to the @login-renew endpoint. http

```
POST /plone/@login-renew HTTP/1.1
Accept: application/json
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
eyJzdWIiOiJhZG1pbiIsImZ1bGxuYW1lIjoiIn0.krI8ep0NQHyYYObtX1lygI4NM2UbHU2qF7kKnx9JFo
```
2.1.4 Invalidating a token (@logout)

The @logout endpoint can be used to invalidate tokens. However by default tokens are not persisted on the server and thus cannot be invalidated. To enable token invalidation, activate the store_tokens option in the PAS plugin. If you need tokens that are valid indefinitely you should also disable the use of Plone’s keyring in the PAS plugin (option use_keyring).

The logout request must contain the existing token in the Authorization header. http

```
POST /plone/@logout HTTP/1.1
Accept: application/json
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
    eyJzdWIiOiJhZG1pbiIsImZ1bGxuYW1lIjoiIn0.krI8ep0NQHyYYObtX1lygI4NM2UbHU12qF7kKnx9JFo
```

curl

```
    'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
    eyJzdWIiOiJhZG1pbiIsImZ1bGxuYW1lIjoiIn0.krI8ep0NQHyYYObtX1lygI4NM2UbHU12qF7kKnx9JFo'
```

httpie

```
    eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
    eyJzdWIiOiJhZG1pbiIsImZ1bGxuYW1lIjoiIn0.
    krI8ep0NQHyYYObtX1lygI4NM2UbHU12qF7kKnx9JFo'
```

python-requests

```
requests.post('http://nohost/plone/@logout', headers={'Accept': 'application/json',
    'Authorization': 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
    eyJzdWIiOiJhZG1pbiIsImZ1bGxuYW1lIjoiIn0.
    krI8ep0NQHyYYObtX1lygI4NM2UbHU12qF7kKnx9JFo
    '})
```

If invalidation succeeds, the server responds with an empty 204 response:

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

2.2 Permissions

In order for a user to use the REST API, the plone.restapi: Use REST API permission is required.

By default, installing the plone.restapi:default profile will assign this permission to the Anonymous role, so everybody is allowed to use the REST API by default.

If you wish to control in more detail which roles are allowed to use the REST API, please assign this permission accordingly.

As well as the plone.restapi: Use REST API permission some of the common Plone permissions are also required, depending on the particular service. For example, retrieving a resource using GET will require View, adding an object using POST will require Add portal content, and so on.

In order to modify/override this behavior, if your custom service class inherits from plone.restapi.services.Service, just override the method check_permission and add your custom checks accordingly.
Explore the API using Postman

To discover the API interactively, using Postman is recommended.

Note
The Chrome-Extension version of Postman is deprecated and it is recommended to use the native app available instead.

3.1 Configuration

To easily follow links returned by request based on the API,

- go to the menu under the wrench icon on the top right
- choose Settings
- activate the option *Retain headers on clicking on links* by selecting *ON*:
This option makes sure, once a HTTP-Header is configured, it will be reused during following requests, if these are initiated by clicking on links resulting from the initial request. This way navigating the structure using the API becomes a snap.

The option Send anonymous usage data to Postman should be deactivated by setting to OFF.

### 3.2 Usage

Choose the suitable HTTP Verb to be used for your request. This can be selected using the Postman HTTP Verb -> GET drop-down menu.

Enter the Object URL of the object that should be the target of a request into the URL field right to the HTTP Verb:

Now set the appropriate HTTP headers.

- The Authorization Header for the authentication related to the right user
- The Accept Header to initiate the right behaviour by the API related to this Request.

To set the Authorization Header, there is a reserved tab, that is responsible to generate the final Header based on the authentication method and username + password.

You have to select

- in the drop-down menu Basic Auth -> the term Basic Auth as the authentication method
- A valid existing user with appropriate permissions
After providing these parameters you can create the resulting Authorization Header and insert it into the prepared request by clicking on Preview Request.

Under the Headers tab you now need to insert in the Accept Header application/json` header as well:

The request is now ready and can be send by clicking on Send button.

The Response of the server is now displayed below the Request. You can easily follow the links on the @id attributes by clicking on them. For every link Postman has prepared another request sharing the same headers that can be send again by licking on the Send button.

Conclusion

You can now explore the whole stucture of your application easily via the API using GET requests.
CHAPTER 4

Content Manipulation

plone.restapi does not only expose content objects via a RESTful API. The API consumer can create, read, update, and delete a content object. Those operations can be mapped to the HTTP verbs POST (Create), GET (Read), PUT (Update) and DELETE (Delete).

Manipulating resources across the network by using HTTP as an application protocol is one of core principles of the REST architectural pattern. This allows us to interact with a specific resource in a standardized way:

<table>
<thead>
<tr>
<th>Verb</th>
<th>URL</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/folder</td>
<td>Creates a new document within the folder</td>
</tr>
<tr>
<td>GET</td>
<td>/folder/{document-id}</td>
<td>Request the current state of the document</td>
</tr>
<tr>
<td>PATCH</td>
<td>/folder/{document-id}</td>
<td>Update the document details</td>
</tr>
<tr>
<td>DELETE</td>
<td>/folder/{document-id}</td>
<td>Remove the document</td>
</tr>
</tbody>
</table>

4.1 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: http

```bash
POST /plone/folder HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

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

curl
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.

A special property during content creation is “UID”, as it requires the user to have the `Manage Portal` permission to set it. Without the permission, the request will fail as Unauthorized.

### 4.1.1 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 representation in the payload:

```json
HTTP/1.1 201 Created
Content-Type: application/json
Location: http://localhost:55001/plone/folder/my-document

{
  "@components": {
    "actions": {
      "@id": "http://localhost:55001/plone/folder/my-document/@actions"
    },
    "breadcrumbs": {
      "@id": "http://localhost:55001/plone/folder/my-document/@breadcrumbs"
    },
    "contextnavigation": {
      "@id": "http://localhost:55001/plone/folder/my-document/@contextnavigation"
    },
    "navigation": {
      "@id": "http://localhost:55001/plone/folder/my-document/@navigation"
    },
    "types": {
      "@id": "http://localhost:55001/plone/folder/my-document/@types"
    },
    "workflow": {
      "@id": "http://localhost:55001/plone/folder/my-document/@workflow"
    }
  }
```

(continues on next page)
4.1.2 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
```
The response body can contain information about why the request failed.

### 4.1.3 Unsuccessful Response (500 Internal Server Error)

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

```plaintext
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.

### 4.1.4 Possible POST Responses

Possible server responses 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)

### 4.1.5 POST Implementation

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

```python
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... [ ]

### 4.2 Reading a Resource with GET

After a successful POST, we can access the resource by sending a GET request to the resource URL: http
GET /plone/folder/my-document HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0

curl


httpie


python-requests

requests.get('http://nohost/plone/folder/my-document', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))

4.2.1 Successful Response (200 OK)

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

HTTP/1.1 200 OK
Content-Type: application/json
{
    "@components": {
        "actions": {
            "@id": "http://localhost:55001/plone/folder/my-document/@actions"
        },
        "breadcrumbs": {
            "@id": "http://localhost:55001/plone/folder/my-document/@breadcrumbs"
        },
        "contextnavigation": {
            "@id": "http://localhost:55001/plone/folder/my-document/@contextnavigation"
        },
        "navigation": {
            "@id": "http://localhost:55001/plone/folder/my-document/@navigation"
        },
        "types": {
            "@id": "http://localhost:55001/plone/folder/my-document/@types"
        },
        "workflow": {
            "@id": "http://localhost:55001/plone/folder/my-document/@workflow"
        }
    },
    "@id": "http://localhost:55001/plone/folder/my-document",
    "@type": "Document",
    "UID": "SomeUUID00000000000000000000000000000005",
    "allow_discussion": false,
    "changeNote": "",
    "contributors": [],
    "created": "1995-07-31T13:45:00",
    "creators": [
        "admin"
    ],

(continues on next page)
For folderish types, their children are automatically included in the response as `items`. To disable the inclusion, add the GET parameter `include_items=false` to the URL.

By default only basic metadata is included. To include additional metadata, you can specify the names of the properties with the `metadata_fields` parameter. See also `Retrieving additional metadata`.

The following example additionally retrieves the UID and Creator: http

```plaintext
GET /plone/folder?metadata_fields=UID&metadata_fields=Creator HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl


httpie

```
```
4.2. Reading a Resource with GET

python-requests

```python
             headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

HTTP/1.1 200 OK
Content-Type: application/json

```json
{
   "@components": {
      "actions": {
         "@id": "http://localhost:55001/plone/folder/@actions"
      },
      "breadcrumbs": {
         "@id": "http://localhost:55001/plone/folder/@breadcrumbs"
      },
      "contextnavigation": {
         "@id": "http://localhost:55001/plone/folder/@contextnavigation"
      },
      "navigation": {
         "@id": "http://localhost:55001/plone/folder/@navigation"
      },
      "types": {
         "@id": "http://localhost:55001/plone/folder/@types"
      },
      "workflow": {
         "@id": "http://localhost:55001/plone/folder/@workflow"
      }
   },
   "@id": "http://localhost:55001/plone/folder",
   "@type": "Folder",
   "UID": "SomeUUID000000000000000000000002",
   "allow_discussion": false,
   "contributors": [],
   "created": "1995-07-31T13:45:00",
   "creators": ["test_user_1_"],
   "description": "This is a folder with two documents",
   "effective": null,
   "exclude_from_nav": false,
   "expires": null,
   "id": "folder",
   "is_folderish": true,
   "items": [
      {
         "@id": "http://localhost:55001/plone/folder/doc1",
         "@type": "Document",
         "Creator": "test_user_1_",
         "UID": "SomeUUID000000000000000000000003",
         "description": "",
         "review_state": "private",
         "title": "A document within a folder"
      },
      {
         "@id": "http://localhost:55001/plone/folder/doc2",
         "@type": "Document",
         "Creator": "test_user_1_"
      }
   ]
}
```
"UID": "SomeUUID000000000000000000000004",
"description": "",
"review_state": "private",
"title": "A document within a folder"
},
{
"@id": "http://localhost:55001/plone/folder/my-document",
"@type": "Document",
"Creator": "admin",
"UID": "SomeUUID000000000000000000000005",
"description": "",
"review_state": "private",
"title": "My Document"
}
],
"items_total": 3,
"language": "",
"layout": "listing_view",
"modified": "1995-07-31T17:30:00",
"nextPreviousEnabled": false,
"next_item": {},
"parent": {
"@id": "http://localhost:55001/plone",
"@type": "Plone Site",
"description": "",
"title": "Plone site"
},
"previous_item": {
"@id": "http://localhost:55001/plone/front-page",
"@type": "Document",
"description": "Congratulations! You have successfully installed Plone.",
"title": "Welcome to Plone"
},
"relatedItems": [],
"review_state": "private",
"rights": "",
"subjects": [],
"title": "My Folder",
"version": "current",
"working_copy": null,
"working_copy_of": null
}

---

**Note:** For folderish types, collections or search results, the results will be *batched* if the size of the resultset exceeds the batch size. See *Batching* for more details on how to work with batched results.

---

### 4.2.2 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
```
4.2.3 GET Implementation

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

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

You can find implementation details in the `plone.restapi.services.content.add.FolderPost` class.

4.2.4 GET Responses

Possible server responses for a GET request are:

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

4.3 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).

If you send the value `null` for a field, the field's content will be deleted and the `missing_value` defined for the field in the schema will be set. Note that this is not possible if the field is `required`, and it only works for Dexterity types, not Archetypes: http

```
PATCH /plone/folder/my-document HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
    "title": "My New Document Title"
}
```

curl
4.3.1 Successful Response (204 No Content)

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

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

4.3.2 Successful Response (200 OK)

You can get the object representation by adding a `Prefer` header with a value of `return=representation` to the PATCH request. In this case, the response will be a **200 OK**: http

```
PATCH /plone/folder/my-document HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Prefer: return=representation
Content-Type: application/json

{
  "title": "My New Document Title"
}
```
requests.patch('http://nohost/plone/folder/my-document', headers={'Accept': 'application/json', 'Content-Type': 'application/json', 'Prefer': 'return=representation'}, json={'title': 'My New Document Title'}, auth=('admin', 'secret'))

HTTP/1.1 200 OK
Content-Type: application/json

{
  "@components": {
    "actions": {
      "@id": "http://localhost:55001/plone/folder/my-document/@actions"
    },
    "breadcrumbs": {
      "@id": "http://localhost:55001/plone/folder/my-document/@breadcrumbs"
    },
    "contextnavigation": {
      "@id": "http://localhost:55001/plone/folder/my-document/@contextnavigation"
    },
    "navigation": {
      "@id": "http://localhost:55001/plone/folder/my-document/@navigation"
    },
    "types": {
      "@id": "http://localhost:55001/plone/folder/my-document/@types"
    },
    "workflow": {
      "@id": "http://localhost:55001/plone/folder/my-document/@workflow"
    }
  },
  "@id": "http://localhost:55001/plone/folder/my-document",
  "@type": "Document",
  "UID": "SomeUUID000000000000000000000000000000005",
  "allow_discussion": false,
  "changeNote": "",
  "contributors": [],
  "created": "1995-07-31T13:45:00",
  "creators": ["admin"],
  "description": "",
  "effective": null,
  "exclude_from_nav": false,
  "expires": null,
  "id": "my-document",
  "is_folderish": false,
  "language": "",
  "layout": "document_view",
  "modified": "1995-07-31T13:45:00",
  "next_item": {},
  "parent": {
    "@id": "http://localhost:55001/plone/folder",
    "@type": "Folder",
    "description": "This is a folder with two documents",
    "review_state": "private",
    "title": "My Folder"
  },
  "previous_item": {
(continues on next page)

### 4.4 Replacing a Resource with PUT

**Note:** PUT is not implemented yet.

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

TODO: Add example.

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).

#### 4.4.1 Successful Update (204 No Content)

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

TODO: Add example.

#### 4.4.2 Unsuccessful Update (409 Conflict)

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

TODO: Add example.
4.4.3 PUT Implementation

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

```python
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...

4.4.4 PUT Responses

Possible server responses for a PUT request are:

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

4.4.5 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.

4.5 Removing a Resource with DELETE

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

```bash
DELETE /plone/folder/my-document HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

```bash
curl
```
A successful response will be indicated by a 204 No Content response:

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

### 4.5.1 DELETE Implementation

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

```python
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…

### 4.5.2 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
4.6 Reordering sub resources

The resources contained within a resource can be reordered using the ordering key using a PATCH request on the container.

Use the obj_id subkey to specify which resource to reorder. The subkey delta can be ‘top’, ‘bottom’, or a negative or positive integer for moving up or down.

Reordering resources within a subset of resources can be done using the subset_ids subkey.

A response 400 BadRequest with a message ‘Client/server ordering mismatch’ will be returned if the value is not in the same order as serverside.

A response 400 BadRequest with a message ‘Content ordering is not supported by this resource’ will be returned if the container does not support ordering.

```python
requests.patch('http://nohost/plone/folder/my-document', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'ordering': {'delta': 'top', 'obj_id': 'item_3', 'subset_ids': ['item_1', 'item_3', 'item5']}, 'auth': ('admin', 'secret')})
```

To rearrange all items in a folderish context use the sort key.

The on subkey defines the catalog index to be sorted on. The order subkey indicates ‘ascending’ or ‘descending’ order of items.
A response 400 BadRequest with a message ‘Content ordering is not supported by this resource’ will be returned if the container does not support ordering. http

`PATCH /plone/folder HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json`

```
{
    "sort": {"on": "modified", "order": "descending"}
}
```

curl

```
```

httpie

```
echo '{
    "sort": {
        "on": "modified",
        "order": "descending"
    }
```

python-requests

```
requests.patch('http://nohost/plone/folder', headers={"Accept": 'application/json', "Content-Type": 'application/json'}, json={'sort': {'on': 'modified', 'order': 'descending'}}, auth=('admin', 'secret'))
```
5.1 Retrieving blocks on a content object

The `plone.restapi.behaviors.IBlocks` has two fields where existing blocks and their data are stored in the object (blocks) and the one where the current layout is stored (blocks_layout). As they are fields in a Dexterity behavior, both fields will be returned in a simple GET as attributes:

```http
GET /plone/my-document HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

The server responds with a `Status 200` and list all stored blocks on that content object:

```json
{
    "@id": "http://localhost:55001/plone/my-document",
    ...
    "blocks_layout": [
    "#title-1",
    "#description-1",
    "#image-1"
    ],
    ...}
```

(continues on next page)
blocks objects will contain the tile metadata and the information to required to render them.

## 5.2 Adding blocks to an object

Storing blocks is done also via a default PATCH content operation:

```json
PATCH /plone/my-document HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{  
  "blocks_layout": [  
    "#title-1",  
    "#description-1",  
    "#image-1"
  ],
  "blocks": {  
    "#title-1": {  
      "@type": "title"
    },
    "#description-1": {  
      "@type": "Description"
    },
    "#image-1": {  
      "@type": "Image",
      "image": "<some random url>"
    }
  }
}
```

## 5.3 Block serializers and deserializers

Practical experience has shown that it is useful to transform, server-side, the value of block fields on inbound (deserialization) and also outbound (serialization) operations. For example, HTML field values are cleaned up using `portal_transforms`, paths in Image blocks are transformed to use `resolveuid` and so on.

It is possible to influence the transformation of block values per block type. For example, to tweak the value stored in Image type block, we can create a new subscriber like:
@implementer(IBlockFieldDeserializationTransformer)
@adapter(IBlocks, IBrowserRequest)
class ImageBlockDeserializeTransformer(object):
    order = 100
    block_type = 'image'

    def __init__(self, context, request):
        self.context = context
        self.request = request

    def __call__(self, value):
        portal = getMultiAdapter((self.context, self.request), name="plone_portal_state" ).portal()
        url = value.get('url', '')
        deserialized_url = path2uid(context=self.context, portal=portal, href=url)
        value['url'] = deserialized_url
        return value

Then register as a subscription adapter:

<subscriber factory=".blocks.ImageBlockDeserializeTransformer"
    provides="plone.restapi.interfaces.IBlockFieldDeserializationTransformer"/>

This would replace the url value to use resolveuid instead of hardcoding the image path.

The block_type attribute needs to match the @type field of the block value. The order attribute is used in sorting the subscribers for the same field. Lower number has higher precedence (is executed first).

On the serialization path, a block value can be tweaked with a similar transformer, for example on an imaginary Database Listing block type:

@implementer(IBlockFieldDeserializationTransformer)
@adapter(IBlocks, IBrowserRequest)
class DatabaseQueryDeserializeTransformer(object):
    order = 100
    block_type = 'database_listing'

    def __init__(self, context, request):
        self.context = context
        self.request = request

    def __call__(self, value):
        value['items'] = db.query(value)  # pseudocode
        return value

Then register as a subscription adapter:

<subscriber factory=".blocks.DatabaseQueryDeserializeTransformer"
    provides="plone.restapi.interfaces.IBlockFieldDeserializationTransformer"/>
5.3.1 Generic block transformers and smart fields

You can create a block transformer that applies to all blocks, by using `None` as the value for `block_type`. The `order` field still applies, though. Using the generic block transformers enables us to create **smart block fields**, which are handled differently. For example, any internal link stored as `url` or `href` in a block value is converted (and stored) as a `resolveuid`-based URL, then resolved back to a full URL on block serialization.

Another **smart field** is the `searchableText` field in a block value. It needs to be a plain text value and it will be used in the `SearchableText` value for the context item.

If you need to store “subblocks” in a block value, you should use the `blocks` smart field (or `data.blocks`), doing so integrates those blocks with the transformers.

5.4 SearchableText indexing for blocks

As the main consumer of `plone.restapi`'s blocks, this functionality is specific to Volto blocks. By default searchable text (for Plone’s `SearchableText` index) is extracted from `text` blocks.

To extract searchable text for other types of blocks, you need to write an adapter that can process that type of block:

```python
@implementer(IBlockSearchableText)
@adapter(IBlocks, IBrowserRequest)
class ImageSearchableText(object):
    def __init__(self, context, request):
        self.context = context
        self.request = request

    def __call__(self, block_value):
        return block_value['alt_text']
```

See `plone.restapi.interfaces.IBlockSearchableText` for details. The `__call__` method needs to return a string, for the text to be indexed.

This adapter needs to be registered as a named adapter, where the name is the same as the block type (its `@type` property from the block value):

```xml
<adapter name="image" factory="/indexers.ImageBlockSearchableText" />
```
History

The @history endpoint exposes history and versioning information on previous versions of the content. Each change or workflow change on a content object or file is listed. It also allows to revert to a previous version of the file.

6.1 Listing the History of a Content Object

Listing versions and history of a resource: http

```plaintext
GET /plone/front-page/@history HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

```bash
curl -i http://nohost/plone/front-page/@history -H 'Accept: application/json' --user admin:secret
```

```bash
httpie
```

```python
requests.get('http://nohost/plone/front-page/@history', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

```plaintext
HTTP/1.1 200 OK
Content-Type: application/json
```

```json
{
  "@id": "http://localhost:55001/plone/front-page/@history/0",
}
```

(continues on next page)
This following fields are returned:

- **action**: the workflow transition id, ‘Edited’ for versioning, or ‘Create’ for initial state.
- **actor**: the user who performed the action. This contains a subobject with the details.
- **comments**: a changenote
- **@id**: link to the content endpoint of this specific version.
- **may_revert**: true if the user has permission to revert.
- **time**: when this action occurred in ISO format.
- **transition_title**: the workflow transition’s title, ‘Edited’ for versioning, or ‘Create’ for initial state.
- **type**: ‘workflow’ for workflow changes, ‘versioning’ for editing, or null for content creation.
- **version**: identifier for this specific version of the resource.

### 6.2 Get a Historical Version

Older versions of a resource can be retrieved by appending `version` to the @history endpoint url. http
6.3 Revert to a Historical Version

Reverting to an older versions of a resource can be done by sending a PATCH request to the @history endpoint and appending the version you want to revert to. http

```bash
PATCH /plone/front-page/@history HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
   "version": 0
}
```

curl

```
```

httpie

```
echo '{
   "version": 0
}' | http PATCH http://nohost/plone/front-page/@history Accept:application/json -a admin:secret
```

python-requests

```python
requests.patch('http://nohost/plone/front-page/@history', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'version': 0}, auth=('admin', 'secret'))
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "message": "Welcome to Plone has been reverted to revision 0."
}
```
Representations of collection-like resources are batched / paginated if the size of the resultset exceeds the batching size:

```
{
    "@id": "http://.../folder/search",
    "batching": {
        "@id": "http://.../folder/search?b_size=10&b_start=20",
        "first": "http://.../plone/folder/search?b_size=10&b_start=0",
        "last": "http://.../plone/folder/search?b_size=10&b_start=170",
        "prev": "http://.../plone/folder/search?b_size=10&b_start=10",
        "next": "http://.../plone/folder/search?b_size=10&b_start=30"
    },
    "items": [
        "...
    ],
    "items_total": 175,
}
```

If the entire resultset fits into a single batch page (as determined by $b_{\text{size}}$), the top-level $\text{batching}$ links will be omitted.

### 7.1 Top-level attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@id</td>
<td>Canonical base URL for the resource, without any batching parameters</td>
</tr>
<tr>
<td>items</td>
<td>Current batch of items / members of the collection-like resource</td>
</tr>
<tr>
<td>items_total</td>
<td>Total number of items</td>
</tr>
<tr>
<td>batching</td>
<td>Batching related navigation links (see below)</td>
</tr>
</tbody>
</table>
7.2 Batching links

If, and only if, the resultset has been batched over several pages, the response body will contain a top-level attribute `batching` that contains batching links. These links that can be used to navigate batches in a Hypermedia fashion:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@id</td>
<td>Link to the current batch page</td>
</tr>
<tr>
<td>first</td>
<td>Link to the first batch page</td>
</tr>
<tr>
<td>prev</td>
<td>Link to the previous batch page (if applicable)</td>
</tr>
<tr>
<td>next</td>
<td>Link to the next batch page (if applicable)</td>
</tr>
<tr>
<td>last</td>
<td>Link to the last batch page</td>
</tr>
</tbody>
</table>

7.3 Parameters

Batching can be controlled with two query string parameters. In order to address a specific batch page, the `b_start` parameter can be used to request a specific batch page, containing `b_size` items starting from `b_start`.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b_size</td>
<td>Batch size (default is 25)</td>
</tr>
<tr>
<td>b_start</td>
<td>First item of the batch</td>
</tr>
</tbody>
</table>

Full example of a batched request and response: http

```
GET /plone/folder/@search?b_size=5&sort_on=path HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

```
curl -i 'http://nohost/plone/folder/@search?b_size=5&sort_on=path' -H 'Accept: application/json' --user admin:secret
```

```
http 'http://nohost/plone/folder/@search?b_size=5&sort_on=path' Accept:application/json -a admin:secret
```

```
requests.get('http://nohost/plone/folder/@search?b_size=5&sort_on=path', headers={
    'Accept': 'application/json'}, auth=('admin', 'secret'))
```

```json
HTTP/1.1 200 OK
Content-Type: application/json

{
  "@id": "http://localhost:55001/plone/folder/@search",
  "batching": {
    "@id": "http://localhost:55001/plone/folder/@search?b_size=5&sort_on=path",
    "first": "http://localhost:55001/plone/folder/@search?b_start=0&b_size=5&sort_on=path",
    "last": "http://localhost:55001/plone/folder/@search?b_start=5&b_size=5&sort_on=path"
  }
}
```
"next": "http://localhost:55001/plone/folder/@search?b_start=5&b_size=5&sort_on=path",
"items": [
    {
      "@id": "http://localhost:55001/plone/folder",
      "@type": "Folder",
      "description": "",
      "review_state": "private",
      "title": "Folder"
    },
    {
      "@id": "http://localhost:55001/plone/folder/doc-1",
      "@type": "Document",
      "description": "",
      "review_state": "private",
      "title": "Document 1"
    },
    {
      "@id": "http://localhost:55001/plone/folder/doc-2",
      "@type": "Document",
      "description": "",
      "review_state": "private",
      "title": "Document 2"
    },
    {
      "@id": "http://localhost:55001/plone/folder/doc-3",
      "@type": "Document",
      "description": "",
      "review_state": "private",
      "title": "Document 3"
    },
    {
      "@id": "http://localhost:55001/plone/folder/doc-4",
      "@type": "Document",
      "description": "",
      "review_state": "private",
      "title": "Document 4"
    }
],
"items_total": 8
Add-on product records can be addressed through the @addons endpoint on the Plone site. In order to address a specific record, the profile id has to be passed as a path segment (e.g. /plone/@addons/plone.session).

Reading or writing addons metadata require the cmf.ManagePortal permission.

## 8.1 Reading add-ons records

Reading a single record: http

```plaintext
GET /plone/@addons/plone.session HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

```
curl -i http://nohost/plone/@addons/plone.session -H 'Accept: application/json' --user admin:secret
```

```
http http://nohost/plone/@addons/plone.session Accept:application/json -a admin:secret
```

```
requests.get('http://nohost/plone/@addons/plone.session', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

Example Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

(continues on next page)
8.2 Listing add-ons records

A list of all add-ons in the portal can be retrieved by sending a GET request to the @addons endpoint: http

```
GET /plone/@addons HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

```
curl -i http://nohost/plone/@addons -H 'Accept: application/json' --user admin:secret
```

```
http http://nohost/plone/@addons Accept:application/json -a admin:secret
```

```
python-requests
```
requests.get('http://nohost/plone/@addons', headers={'Accept': 'application/json'},
auth=('admin', 'secret'))

Response:

HTTP/1.1 200 OK
Content-Type: application/json

{
  "@id": "http://localhost:55001/plone/@addons",
  "items": [
    {
      "description": "Optional plone.session refresh support.",
      "profile_type": "default",
      "upgrade_profiles": {},
      "uninstall_profile_id": "plone.session:uninstall",
      "other_profiles": [],
      "is_installed": "FALSE",
      "id": "plone.session",
      "install_profile": {
        "product": "plone.session",
        "description": "Optional plone.session refresh support.",
        "for": "NULL",
        "title": "Session refresh support",
        "pre_handler": "NULL",
        "version": "1001",
        "type": 2,
        "id": "plone.session:default",
        "post_handler": "NULL"
      },
      "title": "Session refresh support",
      "uninstall_profile": {
        "product": "plone.session",
        "description": "Optional plone.session refresh support. [uninstall]",
        "for": "NULL",
        "title": "Session refresh support [uninstall]",
        "pre_handler": "NULL",
        "type": 2,
        "id": "plone.session:uninstall",
        "post_handler": "NULL"
      },
      "install_profile_id": "plone.session:default",
      "version": "3.7.0",
      "upgrade_info": {}
    }
  ],
  "items_total": 16
}

The following fields are returned:

- @id: hypermedia link to the control panel
- id: the name of the add-on package
- title: the friendly name of the add-on package
- description: description of the add-on
- version: the current version of the add-on
• is_installed: is the add-on installed?
• has_uninstall_profile: does the add-on have an uninstall profile

### 8.3 Installing an addon

An individual addon can be installed by issuing a POST to the given URL: http

```plaintext
POST /plone/@addons/plone.session/install HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```bash
curl -i -X POST http://nohost/plone/@addons/plone.session/install -H 'Accept: application/json' --user admin:secret
```

httpie

```bash
```

python-requests

```python
requests.post('http://nohost/plone/@addons/plone.session/install', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

HTTP/1.1 204 No Content

### 8.4 Uninstalling an addon

An individual addon can be uninstalled by issuing a POST to the given URL: http

```plaintext
POST /plone/@addons/plone.session/uninstall HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```bash
curl -i -X POST http://nohost/plone/@addons/plone.session/uninstall -H 'Accept: application/json' --user admin:secret
```

httpie

```bash
```

python-requests

```python
requests.post('http://nohost/plone/@addons/plone.session/uninstall', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```
8.5 Upgrading an addon

An individual addon can be upgraded by issuing a POST to the given URL: http

```
POST /plone/@addons/plone.session/upgrade HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i -X POST http://nohost/plone/@addons/plone.session/upgrade -H 'Accept: application/json' --user admin:secret
```

httpie

```
```

python-requests

```
requests.post('http://nohost/plone/@addons/plone.session/upgrade', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

HTTP/1.1 204 No Content
Plone offers users to post comments on any content object with plone.app.discussion. Commenting can be enabled globally, for specific content types and for single content objects. When commenting is enabled on your content object, you can retrieve a list of all existing comments, add new comments, reply to existing comments or delete a comment.

### 9.1 Listing Comments

You can list the existing comment on a content object by sending a GET request to the URL of the content object and appending '/@comments': http

```
GET /plone/front-page/@comments HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl
```
curl -i http://nohost/plone/front-page/@comments -H 'Accept: application/json' --user admin:secret
```

httpie
```
http http://nohost/plone/front-page/@comments Accept:application/json -a admin:secret
```

python-requests
```
requests.get('http://nohost/plone/front-page/@comments', headers={'Accept': '<application/json'}, auth=('admin', 'secret'))
```

The server will respond with a **Status 200** and a batched list of all comments:
HTTP/1.1 200 OK
Content-Type: application/json

{
   "@id": "http://localhost:55001/plone/front-page/@comments",
   "items": [
      {
         "@id": "http://localhost:55001/plone/front-page/@comments/1400000000000000",
         "@parent": null,
         "@type": "Discussion Item",
         "author_image": null,
         "author_name": null,
         "author_username": null,
         "comment_id": "1400000000000000",
         "creation_date": "1995-07-31T13:45:00",
         "in_reply_to": null,
         "is_deletable": true,
         "is_editable": true,
         "modification_date": "1995-07-31T17:30:00",
         "text": {
            "data": "Comment 1",
            "mime-type": "text/plain"
         },
         "user_notification": null
      },
      {
         "@id": "http://localhost:55001/plone/front-page/@comments/1400000000000001",
         "@parent": "http://localhost:55001/plone/front-page/@comments/1400000000000000",
         "@type": "Discussion Item",
         "author_image": null,
         "author_name": null,
         "author_username": null,
         "comment_id": "1400000000000001",
         "creation_date": "1995-07-31T13:45:00",
         "in_reply_to": "1400000000000000",
         "is_deletable": true,
         "is_editable": true,
         "modification_date": "1995-07-31T17:30:00",
         "text": {
            "data": "Comment 1.1",
            "mime-type": "text/plain"
         },
         "user_notification": null
      }
   ],
   "items_total": 2
}

These following fields are returned:

- @id: Link to the current endpoint
- items: a list of comments for the current resource
- items_total: the total number of comments for the resource
- batching: batching information

The items attribute returns a list of comments, each comment provides the following fields:
• @id: hyperlink to the comment
• @parent: (optional) the parent comment
• author_name: the full name of the author of this comment
• author_username: the username of the author of this comment
• comment_id: the comment ID uniquely identifies the comment
• in_reply_to: the comment ID of the parent comment
• creation_date: when the comment was placed
• modification_date: when the comment was last updated
• text: contains a ‘mime-type’ and ‘text’ attribute with the text of the comment. Default mime-type is ‘text/plain’.
• user_notification: boolean value to indicate if the author of the comment requested notifications on replies

9.2 Adding a Comment

To add a new comment to a content object, send a POST request to the URL of the content object and append ‘/@comments’ to the URL. The body of the request needs to contain a JSON structure with a ‘text’ attribute that contains the comment text: http

```
POST /plone/front-page/@comments/ HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json
{
  "text": "My comment"
}
```

curl


httpie

echo '{
  "text": "My comment"
}' | http POST http://nohost/plone/front-page/@comments/ Accept:application/json Content-Type:application/json -a admin:secret

python-requests

```
requests.post('http://nohost/plone/front-page/@comments/', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'text': 'My comment'}, auth=('admin', 'secret'))
```

If the creation of the comment has been successful, the server will respond with a 204 No Content status and the URL of the newly created comment in the location header:

```
HTTP/1.1 204 No Content
Location: http://localhost:55001/plone/front-page/@comments/123456
```
# 9.3 Replying to a Comment

To add a direct reply to an existing comment, send a POST request to the URL of the comment you want to reply to. The body of the request needs to contain a JSON structure with a `text` attribute that contains the comment text:

```http
POST /plone/front-page/@comments/123456 HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
   "text": "My reply"
}
```

curl


httpie

echo '{
   "text": "My reply"
}' | http POST http://nohost/plone/front-page/@comments/123456 -a admin:secret

python-requests

```python
requests.post('http://nohost/plone/front-page/@comments/123456', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'text': 'My reply'}, auth=('admin', 'secret'))
```

If the creation of the comment has been successful, the server will respond with a 204 No Content status and the URL of the newly created comment in the location header:

```
HTTP/1.1 204 No Content
Location: http://localhost:55001/plone/front-page/@comments/123456
```

# 9.4 Updating a Comment

**Note:** The permission to update a comment is, by default, only granted to the creator (owner role) of the comment.

An existing comment can be updated by sending a PATCH request to the URL of the comment. The request body needs to contain a JSON structure with at least a `text` attribute:

```http
PATCH /plone/front-page/@comments/123456 HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
}
```

(continues on next page)
curl


httpie

echo '{
  "text": "My NEW comment"
}' | http PATCH http://nohost/plone/front-page/@comments/123456 Accept: application/json -a admin:secret

python-requests

requests.patch('http://nohost/plone/front-page/@comments/123456', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'text': 'My NEW comment'}, auth=('admin', 'secret'))

The server will respond with a **204 No Content** response and a location header with the comment URL when the comment has been updated successfully:

```
HTTP/1.1 204 No Content
Location: http://localhost:55001/plone/front-page/@comments/123456
```

### 9.5 Deleting a Comment

An existing comment can be deleted by sending a DELETE request to the URL of the comment.

**Note:** Deleting a comment will, by default, also delete all existing replies to that comment.

http

```
DELETE /plone/front-page/@comments/123456 HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i -X DELETE http://nohost/plone/front-page/@comments/123456 -H 'Accept: application/json' --user admin:secret
```

httpie

```
http DELETE http://nohost/plone/front-page/@comments/123456 Accept: application/json -a admin:secret
```

python-requests

```
requests.delete('http://nohost/plone/front-page/@comments/123456', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, auth=('admin', 'secret'))
```

**9.5. Deleting a Comment**
requests.delete('http://nohost/plone/front-page/@comments/123456', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))

When the comment has been deleted successfully, the server will respond with a 204 No Content response:

HTTP/1.1 204 No Content
10.1 Copying an object

To copy a content object send a POST request to the /@copy endpoint at the destinations url with the source object specified in the request body. The source object can be specified either by url, path, UID or intid. http

```
POST /plone/@copy HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
  "source": "http://localhost:55001/plone/front-page"
}
```

curl

```
```

httpie

```
```

python-requests

```
requests.post('http://nohost/plone/@copy', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'source': 'http://localhost:55001/plone/front-page'}, auth=('admin', 'secret'))
```
If the copy operation succeeds, the server will respond with status 200 (OK) and return the new and old url of the copied object.

HTTP/1.1 200 OK  
Content-Type: application/json

[  
  {  
    "source": "http://localhost:55001/plone/front-page",  
    "target": "http://localhost:55001/plone/copy_of_front-page"  
  }  
]

10.2 Moving an object

To move a content object send a POST request to the /@move endpoint at the destinations url with the source object specified in the request body. The source object can be specified either by url, path, UID or intid. http

POST /plone/folder/@move HTTP/1.1  
Accept: application/json  
Authorization: Basic YWRtaW46c2VjcmV0  
Content-Type: application/json

{  
  "source": "http://localhost:55001/plone/front-page"  
}

If the move operation succeeds, the server will respond with status 200 (OK) and return the new and old url of the moved object.

HTTP/1.1 200 OK  
Content-Type: application/json

[  
  {  
    "source": "http://localhost:55001/plone/front-page",  
    "target": "http://localhost:55001/plone/copy_of_front-page"  
  }  
]
"target": "http://localhost:55001/plone/folder/front-page"
]
]

## 10.3 Copying/moving multiple objects

Multiple objects can be moved/copied by giving a list of sources. http

```plaintext
POST /plone/@copy HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
  "source": [
    "http://localhost:55001/plone/front-page",
    "http://localhost:55001/plone/newsitem"
  ]
}

curl


httpie

echo '{
}' | http POST http://nohost/plone/@copy Accept:application/json Content-Type:application/json -a admin:secret

python-requests


If the operation succeeds, the server will respond with status 200 (OK) and return the new and old urls for each copied/moved object.

HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "source": "http://localhost:55001/plone/front-page",
    "target": "http://localhost:55001/plone/copy_of_front-page"
  },
  {
    "source": "http://localhost:55001/plone/newsitem",
    "target": "http://localhost:55001/plone/copy_of_newsitem"
  }
]
"source": "http://localhost:55001/plone/newsitem",
"target": "http://localhost:55001/plone/copy_of_newsitem"
}
Expansion

Expansion is a mechanism in plone.restapi to embed additional “components”, such as navigation, breadcrumbs, schema, or workflow within the main content response. This helps the API consumers to avoid unnecessary requests.

Say you want to show a document in Plone together with the breadcrumbs and a workflow switcher. Instead of doing three individual requests, you can just expand the breadcrumbs and the workflow “components” within the document GET request.

The list of expandable components is listed in the “@components” attribute in the response of any content GET request:

```
GET  /plone/front-page HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0

{
    "@id": "http://localhost:55001/plone/front-page",
    "@type": "Document",
    "@components": [ 
        {"@id": "http://localhost:55001/plone/front-page/@actions"},
        {"@id": "http://localhost:55001/plone/front-page/@breadcrumbs"},
        {"@id": "http://localhost:55001/plone/front-page/@navigation"},
        {"@id": "http://localhost:55001/plone/front-page/@types"},
        {"@id": "http://localhost:55001/plone/front-page/@workflow"},
        ...
    ],
    "UID": "1f699ff4aa110e45afb1a502f75f7ec33",
    "title": "Welcome to Plone",
    ...
}
```

Request (unexpanded): http

```
GET  /plone/front-page HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

Request (expanded): http
curl

curl -i http://nohost/plone/front-page -H 'Accept: application/json' --user admin:secret

httpie

http http://nohost/plone/front-page Accept:application/json -a admin:secret

python-requests

requests.get('http://nohost/plone/front-page', headers={'Accept': 'application/json'},
             auth=('admin', 'secret'))

Response (unexpanded):

HTTP/1.1 200 OK
Content-Type: application/json

{
    "@components": {
        "actions": {
            "@id": "http://localhost:55001/plone/front-page/@actions"
        },
        "breadcrumbs": {
            "@id": "http://localhost:55001/plone/front-page/@breadcrumbs"
        },
        "contextnavigation": {
            "@id": "http://localhost:55001/plone/front-page/@contextnavigation"
        },
        "navigation": {
            "@id": "http://localhost:55001/plone/front-page/@navigation"
        },
        "types": {
            "@id": "http://localhost:55001/plone/front-page/@types"
        },
        "workflow": {
            "@id": "http://localhost:55001/plone/front-page/@workflow"
        }
    },
    "@id": "http://localhost:55001/plone/front-page",
    "@type": "Document",
    "UID": "SomeUUID0000000000000000000001",
    "allow_discussion": false,
    "changeNote": "",
    "contributors": [],
    "created": "1995-07-31T13:45:00",
    "creators": [
        "test_user_1"
    ],
    "description": "Congratulations! You have successfully installed Plone."
    "effective": null,
    "exclude_from_nav": false,
    "expires": null,
    "id": "front-page",
    "is_folderish": false,
    "language": "",
    "layout": "document_view",

(continues on next page)
In order to expand and embed one or more components, use the `expand` GET parameter and provide either a single component or a comma-separated list of the components you want to embed. Say you want to expand the `breadcrumbs` component:

```plaintext
GET /plone/front-page?expand=breadcrumbs HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```shell
```

httpie

```shell
```

python-requests

```python
requests.get('http://nohost/plone/front-page?expand=breadcrumbs', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

Response (breadcrumbs expanded):

```
HTTP/1.1 200 OK
Content-Type: application/json
```
plone.restapi Documentation, Release 1.0a1

```json
{
    "@components": {
        "actions": {
            "@id": "http://localhost:55001/plone/front-page/@actions"
        },
        "breadcrumbs": {
            "@id": "http://localhost:55001/plone/front-page/@breadcrumbs",
            "items": [
                {
                    "@id": "http://localhost:55001/plone/front-page",
                    "title": "Welcome to Plone"
                }
            ],
            "root": "http://localhost:55001/plone"
        },
        "contextnavigation": {
            "@id": "http://localhost:55001/plone/front-page/@contextnavigation"
        },
        "navigation": {
            "@id": "http://localhost:55001/plone/front-page/@navigation"
        },
        "types": {
            "@id": "http://localhost:55001/plone/front-page/@types"
        },
        "workflow": {
            "@id": "http://localhost:55001/plone/front-page/@workflow"
        }
    },
    "@id": "http://localhost:55001/plone/front-page",
    "@type": "Document",
    "UID": "SomeUUID000000000000000000000001",
    "allow_discussion": false,
    "changeNote": "",
    "contributors": [],
    "created": "1995-07-31T13:45:00",
    "creators": [
        "test_user_1_"
    ],
    "description": "Congratulations! You have successfully installed Plone.",
    "effective": null,
    "exclude_from_nav": false,
    "expires": null,
    "id": "front-page",
    "is_folderish": false,
    "language": "",
    "layout": "document_view",
    "modified": "1995-07-31T13:45:00",
    "next_item": {},
    "parent": {
        "@id": "http://localhost:55001/plone",
        "@type": "Plone Site",
        "description": "",
        "title": "Plone site"
    },
    "previous_item": {},
    "relatedItems": [],
    "review_state": "private",
}
```

(continues on next page)
Here is an example of a request that expands all possible expansions: http

GET /plone/front-page?expand=actions,breadcrumbs,navigation,workflow,types HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0

curl

curl -i 'http://nohost/plone/front-page?expand=actions,breadcrumbs,navigation,workflow,types' -H 'Accept: application/json' --user admin:secret

httpie

http 'http://nohost/plone/front-page?expand=actions,breadcrumbs,navigation,workflow,types' Accept:application/json -a admin:secret

python-requests

requests.get('http://nohost/plone/front-page?expand=actions,breadcrumbs,navigation,workflow,types', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))

And the response:

HTTP/1.1 200 OK
Content-Type: application/json

{  
    "@components": {  
        "actions": {  
            "document_actions": [],  
            "object": [  
                {  
                    "icon": "",
                    "id": "view",
                    "title": "View"
                },  
                {  
                    "icon": "",
                    "id": "edit",
                    "title": "Edit"
                }
            ]
        },  
        "breadcrumbs": [],  
        "content_type": "text/plain",
        "data": "If you're seeing this instead of the web site you were expecting, the owner of this web site has just installed Plone. Do not contact the Plone Team or the Plone mailing lists about this.",
        "encoding": "utf-8",
        "icon": "",
        "id": "view",
        "title": "Welcome to Plone",
        "version": "current",
        "versioning_enabled": true,
        "working_copy": null,
        "working_copy_of": null
    }
}
"title": "Edit",
},
{
"icon": 
"id": "folderContents",
"title": "Contents"
},
{
"icon": 
"id": "history",
"title": "History"
},
{
"icon": 
"id": "local_roles",
"title": "Sharing"
}
],
"object_buttons": [
{
"icon": 
"id": "cut",
"title": "Cut"
},
{
"icon": 
"id": "copy",
"title": "Copy"
},
{
"icon": 
"id": "delete",
"title": "Delete"
},
{
"icon": 
"id": "rename",
"title": "Rename"
},
{
"icon": 
"id": "redirection",
"title": "URL Management"
}
],
"portal_tabs": [
{
"icon": 
"id": "index_html",
"title": "Home"
}
],
"site_actions": [
{
"icon": 
"id": "sitemap",
"title": "Site Map"
}
},
{
    "@id": "http://localhost:55001/plone/front-page",
    "description": "Congratulations! You have successfully installed Plone."
    "items": [],
    "review_state": "private",
    "title": "Welcome to Plone"
}
],
"types": [
{
    "@id": "http://localhost:55001/plone/@types/Collection",
    "addable": false,
    "title": "Collection"
},
{
    "@id": "http://localhost:55001/plone/@types/DXTestDocument",
    "addable": false,
    "title": "DX Test Document"
},
{
    "@id": "http://localhost:55001/plone/@types/Event",
    "addable": false,
    "title": "Event"
},
{
    "@id": "http://localhost:55001/plone/@types/File",
    "addable": false,
    "title": "File"
},
{
    "@id": "http://localhost:55001/plone/@types/Folder",
    "addable": false,
    "title": "Folder"
},
{
    "@id": "http://localhost:55001/plone/@types/Image",
    "addable": false,
    "title": "Image"
},
{
    "@id": "http://localhost:55001/plone/@types/Link",
    "addable": false,
    "title": "Link"
},
{
    "@id": "http://localhost:55001/plone/@types/News Item",
    "addable": false,
    "title": "News Item"
},
{
    "@id": "http://localhost:55001/plone/@types/Document",
    "addable": false,
    "title": "Page"
}]} (continues on next page)
"workflow": {
  "@id": "http://localhost:55001/plone/front-page/@workflow",
  "history": [  
    {
      "action": null,
      "actor": "test_user_1_",
      "comments": ",",
      "review_state": "private",
      "time": "1995-07-31T17:30:00",
      "title": "Private"
    }
  ],
  "state": {
    "id": "private",
    "title": "Private"
  },
  "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"
    }
  ]
},
"@id": "http://localhost:55001/plone/front-page",
"@type": "Document",
"UID": "SomeUUID000000000000000000000001",
"allow_discussion": false,
"changeNote": ",",
"contributors": [],
"created": "1995-07-31T13:45:00",
"creators": [
  "test_user_1_"
],
"description": "Congratulations! You have successfully installed Plone.",
"effective": null,
"exclude_from_nav": false,
"expires": null,
"id": "front-page",
"is_folderish": false,
"language": "",
"layout": "document_view",
"modified": "1995-07-31T17:30:00",
"next_item": {},
"parent": {
  "@id": "http://localhost:55001/plone",
  "@type": "Plone Site",
  "description": "",
  "title": "Plone site"
},
"previous_item": {},
"relatedItems": [],
"review_state": "private",  
(continues on next page)
If you're seeing this instead of the web site you were expecting, the owner of this web site has just installed Plone. Do not contact the Plone Team or the Plone mailing lists about this.
Plone has the concept of configurable actions (called “portal_actions”). Each action defines an id, a title, the required permissions and a condition that will be checked to decide if the action will be available for a user. Actions are sorted by categories.

Actions can be used to build UI elements that adapt to the available actions. An example is the Plone toolbar where the “object_tabs” (view, edit, folder contents, sharing) and the “user_actions” (login, logout, preferences) are used to display the user only the actions that are allowed for the currently logged in user.

The available actions for the currently logged in user can be retrieved by calling the @actions endpoint on a specific context. This also works for not authenticated users.

### 12.1 Listing available actions

To list the available actions, send a GET request to the ‘@actions’ endpoint on a specific content object: http

```plaintext
GET /plone/@actions HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

**curl**

```
curl -i http://nohost/plone/@actions -H 'Accept: application/json' --user admin:secret
```

**httpie**

```
http http://nohost/plone/@actions Accept:application/json -a admin:secret
```

**python-requests**

```python
requests.get('http://nohost/plone/@actions', headers={'Accept': 'application/json'},
             auth=('admin', 'secret'))
```
The server will respond with a 200 OK status code. The JSON response contains the available actions categories (object, object_buttons, user) on the top level. Each category contains a list of the available actions in that category:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "object": [
        {
            "icon": "",
            "id": "view",
            "title": "View"
        },
        {
            "icon": "",
            "id": "edit",
            "title": "Edit"
        },
        {
            "icon": "",
            "id": "folderContents",
            "title": "Contents"
        },
        {
            "icon": "",
            "id": "history",
            "title": "History"
        },
        {
            "icon": "",
            "id": "local_roles",
            "title": "Sharing"
        }
    ],
    "object_buttons": [
        {
            "icon": "",
            "id": "cut",
            "title": "Cut"
        },
        {
            "icon": "",
            "id": "copy",
            "title": "Copy"
        },
        {
            "icon": "",
            "id": "delete",
            "title": "Delete"
        },
        {
            "icon": "",
            "id": "rename",
            "title": "Rename"
        }
    ],
    "user": [
    ]
}
```
"icon": "",
"id": "preferences",
"title": "Preferences"
},
{
"icon": "",
"id": "plone_setup",
"title": "Site Setup"
},
{
"icon": "",
"id": "logout",
"title": "Log out"
}]
}

If you want to limit the categories that are returned, pass one or more parameters categories:list, i.e. @action? categories:list=object&categories:list=user.
CHAPTER 13

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: http

GET /plone/front-page/@workflow HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0

curl

```bash
curl -i http://nohost/plone/front-page/@workflow -H 'Accept: application/json' --user admin:secret
```

httpie

```bash
```

python-requests

```python
requests.get('http://nohost/plone/front-page/@workflow', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

HTTP/1.1 200 OK
Content-Type: application/json

```json
{
    "@id": "http://localhost:55001/plone/front-page/@workflow",
    "history": [null,
        "actor": "test_user_1_",
```

(continues on next page)
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: http

```
POST /plone/front-page/@workflow/publish HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0


httpie


python-requests

requests.post('http://nohost/plone/front-page/@workflow/publish', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

HTTP/1.1 200 OK
Content-Type: application/json

```json
{
    "action": "publish",
    "actor": "admin",
    "comments": 
    "review_state": "published",
    "time": "1995-07-31T18:30:00",
    "title": "Published with accent é"
}
```
We can also also change the state recursively for all contained items, provide a comment and set effective and expiration dates:

```
POST /plone/folder/@workflow/publish HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
    "comment": "Publishing my folder...",
    "effective": "2018-01-21T08:00:00",
    "expires": "2019-01-21T08:00:00",
    "include_children": true
}
```

curl

curl -i -X POST http://nohost/plone/folder/@workflow/publish -H 'Accept: application/json' -H 'Content-Type: application/json' --data-raw '{"comment": "Publishing my folder...", "effective": "2018-01-21T08:00:00", "expires": "2019-01-21T08:00:00", "include_children": true}' --user admin:secret

httpie

echo '{
    "comment": "Publishing my folder...",
    "effective": "2018-01-21T08:00:00",
    "expires": "2019-01-21T08:00:00",
    "include_children": true

python-requests

```
requests.post('http://nohost/plone/folder/@workflow/publish', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'comment': 'Publishing my folder...', 'effective': '2018-01-21T08:00:00', 'expires': '2019-01-21T08:00:00', 'include_children': True}, auth=('admin', 'secret'))
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "action": "publish",
    "actor": "admin",
    "comments": "Publishing my folder...",
    "review_state": "published",
    "time": "1995-07-31T18:30:00",
    "title": "Published with accent \u00e9"
}
```
CHAPTER 14

Working Copy

Note: This is only available on Plone 5.

Plone has the “Working copy” feature provided by the core package plone.app.iterate. It allows the users to create a working copy of a (published or live) content object and work with it until it’s ready to be published without having to edit the original object.

This process has several steps of its life cycle:

14.1 Create working Copy (aka Check-out)

The user initiates the process and creates a “working copy” by “checking out” the content: http

```
POST /plone/document/@workingcopy HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl


httpie

```
```

python-requests

```
requests.post('http://nohost/plone/document/@workingcopy', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```
and receives the response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: http://localhost:55001/plone/document
{
}
```

### 14.2 Get the working copy

A working copy has been created and can be accessed querying the content: http

```
GET /plone/document/@workingcopy HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

`curl`

```
```

`httpie`

```
```

`python-requests`

```
requests.get('http://nohost/plone/document/@workingcopy', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

and receives the response:

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "working_copy": {
    "created": "1995-07-31T13:45:00",
    "creator_name": "admin",
    "creator_url": "http://localhost:55001/plone/author/admin",
    "title": "Test document"
  },
  "working_copy_of": null
}
```

the GET content of any object, also states the location of the working copy, if any (working_copy). http

```
GET /plone/document HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl
14.2. Get the working copy

```bash
```

```bash
httpie

```

```python
requests.get('http://nohost/plone/document', headers={'Accept': 'application/json'},
             auth=('admin', 'secret'))
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "components": {
    "actions": {
      "@id": "http://localhost:55001/plone/document/@actions"
    },
    "breadcrumbs": {
      "@id": "http://localhost:55001/plone/document/@breadcrumbs"
    },
    "contextnavigation": {
      "@id": "http://localhost:55001/plone/document/@contextnavigation"
    },
    "navigation": {
      "@id": "http://localhost:55001/plone/document/@navigation"
    },
    "types": {
      "@id": "http://localhost:55001/plone/document/@types"
    },
    "workflow": {
      "@id": "http://localhost:55001/plone/document/@workflow"
    }
  },
  "@id": "http://localhost:55001/plone/document",
  "@type": "Document",
  "UID": "SomeUUID000000000000000000000001",
  "allow_discussion": false,
  "contributors": [],
  "created": "1995-07-31T13:45:00",
  "creators": [
    "test_user_1"
  ],
  "description": "",
  "effective": null,
  "exclude_from_nav": false,
  "expires": null,
  "id": "document",
  "is_folderish": false,
  "language": "",
  "layout": "document_view",
  "modified": "1995-07-31T17:30:00",
  "next_item": {
    "@type": "Document",
    "description": "",
  }...
```

(continues on next page)
"title": "Test document",
"parent": {
  "@id": "http://localhost:55001/plone",
  "@type": "Plone Site",
  "description": "",
  "title": "Plone site"
},
"previous_item": {},
"relatedItems": [],
"review_state": "private",
"rights": "",
"subjects": [],
"table_of_contents": null,
"text": null,
"title": "Test document",
"version": "current",
"working_copy": {
  "created": "1995-07-31T13:45:00",
  "creator_name": "admin",
  "creator_url": "http://localhost:55001/plone/author/admin",
  "title": "Test document"
},
"working_copy_of": null
}

the GET content of any a working copy also returns the original (working_copy_of): http

```
GET /plone/copy_of_document HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl


httpie

```
```

python-requests

```
requests.get('http://nohost/plone/copy_of_document', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "@components": {
    "actions": {
      "@id": "http://localhost:55001/plone/copy_of_document/@actions"
    },
    "breadcrumbs": {
```
14.2. Get the working copy

"contextnavigation": {
  "@id": "http://localhost:55001/plone/copy_of_document/@contextnavigation"
},
"navigation": {
  "@id": "http://localhost:55001/plone/copy_of_document/@navigation"
},
"types": {
  "@id": "http://localhost:55001/plone/copy_of_document/@types"
},
"workflow": {
},
"@type": "Document",
"UID": "SomeUUID00000000000000000000000002",
"allow_discussion": false,
"contributors": [],
"created": "1995-07-31T13:45:00",
"creators": [
  "test_user_1_"
],
"description": "",
"effective": null,
"exclude_from_nav": false,
"expires": null,
"id": "copy_of_document",
"is_folderish": false,
"language": "",
"layout": "document_view",
"modified": "1995-07-31T13:45:00",
"next_item": {},
"parent": {
  "@id": "http://localhost:55001/plone",
  "@type": "Plone Site",
  "description": "",
  "title": "Plone site"
},
"previous_item": {
  "@id": "http://localhost:55001/plone/document",
  "@type": "Document",
  "description": "",
  "title": "Test document"
},
"relatedItems": [],
"review_state": "private",
"rights": "",
"subjects": [],
"table_of_contents": null,
"text": null,
"title": "Test document",
"version": "current",
"working_copy": {
  "created": "1995-07-31T13:45:00",
(continues on next page)
"creator_name": "admin",
"creator_url": "http://localhost:55001/plone/author/admin",
"title": "Test document"
},
"working_copy_of": {
"@id": "http://localhost:55001/plone/document",
"title": "Test document"
}
}

## 14.3 Check-in

Once the user has finished editing the working copy and wants to update the original with the changes in there, or “check-in” the working copy. http

```plaintext
PATCH /plone/copy_of_document/@workingcopy HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```
curl

```bash
curl -i -X PATCH http://nohost/plone/copy_of_document/@workingcopy -H 'Accept:
˓→application/json' --user admin:secret
```

httpie

```bash
http PATCH http://nohost/plone/copy_of_document/@workingcopy Accept:application/json -
˓→a admin:secret
```

python-requests

```python
requests.patch('http://nohost/plone/copy_of_document/@workingcopy', headers={'Accept
˓→': 'application/json'}, auth=('admin', 'secret'))
```

and receives the response:

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

The working copy is deleted afterwards as a result of this process. The PATCH can also be issued in the original (baseline) object.

## 14.4 Delete the working copy (cancel check-out)

If you want to cancel the checkout and delete the working copy (in both the original and the working copy): http

```plaintext
DELETE /plone/copy_of_document/@workingcopy HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```
curl
When a working copy is deleted using the “normal” delete action, it also deletes the relation and cancels the check-out, but that is handled by `plone.app.iterate` internals.
Locking is a mechanism to prevent users from accidentally overriding each others changes. When a user edits a content object in Plone, the object is locked until the user hits the save or cancel button. If a second user tries to edit the object at the same time, she will see a message that this object is locked.

### 15.1 Locking an object

To lock an object send a POST request to the `/@lock` endpoint that is available on any content object in Plone:

```plaintext
POST /plone/front-page/@lock HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i -X POST http://nohost/plone/front-page/@lock -H 'Accept: application/json' --user admin:secret
```

httpie

```
http POST http://nohost/plone/front-page/@lock Accept:application/json -a admin:secret
```

python-requests

```python
requests.post('http://nohost/plone/front-page/@lock', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

If the lock operation succeeds, the server will respond with status **200 OK** and return various information about the lock including the lock token. The token is needed in later requests to update the locked object.

```plaintext
HTTP/1.1 200 OK
Content-Type: application/json
```

(continues on next page)
By default, locks are stealable. That means that another user can unlock the object. If you want to create a non-stealable lock, pass "stealable": false in the request body.

To create a lock with a non-default timeout, you can pass the the timeout value in seconds in the request body.

The following example creates a non-stealable lock with a timeout of 1h. http

```
POST /plone/front-page/@lock HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
    "stealable": false,
    "timeout": 3600
}
```

curl

curl -i -X POST http://nohost/plone/front-page/@lock -H 'Accept: application/json' -H 'Content-Type: application/json' --data-raw '{"stealable": false, "timeout": 3600}' --user admin:secret

httpie

echo '{
    "stealable": false,
    "timeout": 3600
}' | http POST http://nohost/plone/front-page/@lock Accept:application/json Content-Type:application/json -a admin:secret

python-requests

```
requests.post('http://nohost/plone/front-page/@lock', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'stealable': False, 'timeout': 3600}, auth=('admin', 'secret'))
```

The server responds with status 200 OK and returns the lock information.
15.2 Unlocking an object

To unlock an object send a POST request to the `/unlock` endpoint. http

```bash
POST /plone/front-page/@unlock HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0

curl

curl -i -X POST http://nohost/plone/front-page/@unlock -H 'Accept: application/json' --user admin:secret

httpie

http POST http://nohost/plone/front-page/@unlock Accept:application/json -a admin:secret

python-requests

requests.post('http://nohost/plone/front-page/@unlock', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))

The server responds with status 200 OK and returns the lock information.

```bash
HTTP/1.1 200 OK
Content-Type: application/json

```json
{
    "locked": false,
    "stealable": true
}
```
15.4 Getting lock information

To find out if an object is locked or to get information about the current lock you can send a GET request to the @lock endpoint. http

```
GET /plone/front-page/@lock HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i http://nohost/plone/front-page/@lock -H 'Accept: application/json' --user admin:secret
```

httpie

```
http http://nohost/plone/front-page/@lock Accept:application/json -a admin:secret
```

python-requests

```
requests.get('http://nohost/plone/front-page/@lock', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

The server responds with status `200 OK` and returns the information about the lock.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "locked": false,
}
```

(continues on next page)
"stealable": true

15.5 Updating a locked object

To update a locked object with a PATCH request, you have to provide the lock token with the `Lock-Token` header.

```
PATCH /plone/front-page HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Lock-Token: 0.684672730996-0.25195226375-00105A989226:1477076400.000
Content-Type: application/json

{
  "title": "New Title"
}
```

curl

```
```

httpie

```
echo '{
  "title": "New Title"
}' | http PATCH http://nohost/plone/front-page Accept:application/json Content-Type:application/json Lock-Token:0.684672730996-0.25195226375-00105A989226:1477076400.000 -a admin:secret
```

python-requests

```
requests.patch('http://nohost/plone/front-page', headers={'Accept': 'application/json', 'Content-Type': 'application/json', 'Lock-Token': '0.684672730996-0.25195226375-00105A989226:1477076400.000'}, json={'title': 'New Title'}, auth=('admin', 'secret'))
```
Plone comes with a sophisticated user management system that allows to assign users and groups with global roles and permissions. Sometimes this in not enough though and you might want to give users the permission to access or edit a specific part of your website or a specific content object. This is where local roles (located in the Plone sharing tab) come in handy.

### 16.1 Retrieving Local Roles

In plone.restapi, the representation of any content object will include a hypermedia link to the local role / sharing information in the `sharing` attribute:

```plaintext
GET /plone/folder HTTP/1.1
Accept: application/json
```

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

{
  "@id": "http://localhost:55001/plone/folder",
  "@type": "Folder",
  ...
  "sharing": {
    "@id": "http://localhost:55001/plone/folder/@sharing",
    "title": "Sharing",
  }
}
```

The sharing information of a content object can also be directly accessed by appending `/@sharing` to the GET request to the URL of a content object. E.g. to access the sharing information for a top-level folder, do: http

```plaintext
GET /plone/folder/@sharing HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```
The `available_roles` property contains the list of roles that can be managed via the sharing page. It contains dictionaries with the role ID and its translated `title` (as it appears on the sharing page).
16.2 Searching for principals

Users and/or groups without a sharing entry can be found by appending the argument search to the query string. ie

?search=admin. Global roles are marked with the string "global". Inherited roles are marked with the string "acquired".

http

```
GET /plone/folder/doc/@sharing?search=admin HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i 'http://nohost/plone/folder/doc/@sharing?search=admin' -H 'Accept: 
application/json' --user admin:secret
```

httpie

```
```

python-requests

```
requests.get('http://nohost/plone/folder/doc/@sharing?search=admin', headers={'Accept ': 'application/json'}, auth=('admin', 'secret'))
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "available_roles": [
    {
      "id": "Contributor",
      "title": "Can add"
    },
    {
      "id": "Editor",
      "title": "Can edit"
    },
    {
      "id": "Reader",
      "title": "Can view"
    },
    {
      "id": "Reviewer",
      "title": "Can review"
    }
  ],
  "entries": [
    {
      "id": "Administrators",
      "login": null,
      "roles": {
        "Contributor": false,
        "Editor": false,
        "Reader": false,
        "Reviewer": false
      }
    }
  ]
}
```

(continues on next page)
16.3 Updating Local Roles

You can update the ‘sharing’ information by sending a POST request to the object URL and appending /@sharing, e.g. /plone/folder/@sharing. E.g. say you want to give the AuthenticatedUsers group the Reader local role for a folder: http

```
POST /plone/folder/@sharing HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```
Content-Type: application/json

```json
{
    "entries": [
        {
            "id": "AuthenticatedUsers",
            "roles": {
                "Contributor": false,
                "Editor": false,
                "Reader": true,
                "Reviewer": true
            },
            "type": "user"
        }
    ],
    "inherit": true
}
```

curl

curl -i -X POST http://nohost/plone/folder/@sharing -H 'Accept: application/json' -H 'Content-Type: application/json' --data-raw '{"entries": [{"type": "user", "id": "AuthenticatedUsers", "roles": {"Contributor": false, "Reviewer": true, "Editor": false, "Reader": true}}], "inherit": true}' --user admin:secret

httpie

echo '{
    "entries": [
        {
            "id": "AuthenticatedUsers",
            "roles": {
                "Contributor": false,
                "Editor": false,
                "Reader": true,
                "Reviewer": true
            },
            "type": "user"
        }
    ],
    "inherit": true
}' | http POST http://nohost/plone/folder/@sharing Accept:application/json Content-Type:application/json -a admin:secret

python-requests

```python
requests.post('http://nohost/plone/folder/@sharing', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'entries': [{'type': 'user', 'id': 'AuthenticatedUsers', 'roles': {'Contributor': False, 'Reviewer': True, 'Editor': False, 'Reader': True}}], 'inherit': True}, auth=('admin', 'secret'))
```

HTTP/1.1 204 No Content

16.3. Updating Local Roles
Registry records can be addressed through the \texttt{@registry} endpoint on the Plone site. In order to address a specific record, the fully qualified dotted name of the registry record has to be passed as a path segment (e.g. \texttt{/plone/@registry/my.record}).

Reading or writing registry records require the \texttt{cmf.ManagePortal} permission.

### 17.1 Reading registry records

Reading a single record: http

\begin{verbatim}
GET /plone/@registry/plone.app.querystring.field.path.title HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
curl

curl -i http://nohost/plone/@registry/plone.app.querystring.field.path.title -H
    'Accept: application/json' --user admin:secret
httpie

http http://nohost/plone/@registry/plone.app.querystring.field.path.title
    --Accept:application/json -a admin:secret

python-requests

requests.get('http://nohost/plone/@registry/plone.app.querystring.field.path.title',
    headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
\end{verbatim}

Example Response:
17.2 Listing registry records

The registry records listing uses a batched method to access all registry records. See Batching for more details on how to work with batched results.

The output per record contains the following fields: name: The record’s fully qualified dotted name. value: The record’s value. This is the same as GETting @registry/name. http

```
GET /plone/@registry HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i http://nohost/plone/@registry -H 'Accept: application/json' --user admin:secret
```

httpie

```
http http://nohost/plone/@registry Accept:application/json -a admin:secret
```

python-requests

```
requests.get('http://nohost/plone/@registry', headers={'Accept': 'application/json'},
              auth=('admin', 'secret'))
```

Example Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "@id": "http://localhost:55001/plone/@registry",
   "batching": {
      "@id": "http://localhost:55001/plone/@registry",
      "first": "http://localhost:55001/plone/@registry?b_start=0",
      "last": "http://localhost:55001/plone/@registry?b_start=1775",
      "next": "http://localhost:55001/plone/@registry?b_start=25"
   },
   "items": [
      {
         "name": "Products.CMFPPlone.i18nl10n.override_dateformat.Enabled",
         "schema": {
            "properties": {
               "description": "Override the translation machinery",
               "factory": "Yes/No",
               "title": "Enabled",
               "type": "boolean"
            }
         }
      }
   ]
}
```

(continues on next page)
"value": false
},
{
 "name": "Products.CMFPlone.i18nl10n.override_dateformat.date_format_long",
 "schema": {
   "properties": {
     "description": "Default value: %Y-%m-%d %H:%M (2038-01-19 03:14)",
     "factory": "Text line (String)",
     "title": "old ZMI property: localLongTimeFormat",
     "type": "string"
   }
  },
  "value": "%Y-%m-%d %H:%M"
 },
{
 "name": "Products.CMFPlone.i18nl10n.override_dateformat.date_format_short",
 "schema": {
   "properties": {
     "description": "Default value: %Y-%m-%d (2038-01-19)",
     "factory": "Text line (String)",
     "title": "old ZMI property: localTimeFormat",
     "type": "string"
   }
  },
  "value": "%Y-%m-%d"
 },
{
 "name": "Products.CMFPlone.i18nl10n.override_dateformat.time_format",
 "schema": {
   "properties": {
     "description": "Default value: %H:%M (03:14)",
     "factory": "Text line (String)",
     "title": "old ZMI property: localTimeOnlyFormat",
     "type": "string"
   }
  },
  "value": "%H:%M"
 },
{
 "name": "Products.CMFPlone.interfaces.syndication.ISiteSyndicationSettings.allowed",
 "schema": {
   "properties": {
     "default": true,
     "description": "Allow syndication for collections and folders on site.",
     "factory": "Yes/No",
     "title": "Allowed",
     "type": "boolean"
   }
  },
  "value": true
 },
{
 "name": "Products.CMFPlone.interfaces.syndication.ISiteSyndicationSettings.allowed_feed_types",
 "schema": {
   "properties": {
}}
"additionalItems": true,
"default": [
  "RSS|RSS 1.0",
  "rss.xml|RSS 2.0",
  "atom.xml|Atom",
  "itunes.xml|iTunes"
],
"description": "Separate view name and title by '|'",
"factory": "Tuple",
"items": {
  "description": "",
  "factory": "Text line (String)"
},
"title": "Allowed Feed Types",
"type": "array",
"uniqueItems": true
},
"value": [
  "RSS|RSS 1.0",
  "rss.xml|RSS 2.0",
  "atom.xml|Atom",
  "itunes.xml|iTunes"
]}
),

{ "name": "Products.CMFPlone.interfaces.syndication.ISiteSyndicationSettings.default_enabled",
 "schema": {
  "properties": {
    "default": false,
    "description": "If syndication should be enabled by default for all folders and collections.",
    "factory": "Yes/No",
    "title": "Enabled by default",
    "type": "boolean"
  }
 },
 "value": false
},

{ "name": "Products.CMFPlone.interfaces.syndication.ISiteSyndicationSettings.max_items",
 "schema": {
  "properties": {
    "default": 15,
    "description": "Maximum number of items that will be syndicated.",
    "factory": "Integer",
    "minimum": 1,
    "title": "Maximum items",
    "type": "integer"
  }
 },
 "value": 15
}


```json
{
    "name": "Products.CMFPlone.interfaces.syndication.ISiteSyndicationSettings.render_body",
    "schema": {
        "properties": {
            "default": false,
            "description": "If body text available for item, render it, otherwise use description.",
            "factory": "Yes/No",
            "title": "Render Body",
            "type": "boolean"
        }
    },
    "value": false
},
{
    "name": "Products.CMFPlone.interfaces.syndication.ISiteSyndicationSettings.search_rss_enabled",
    "schema": {
        "properties": {
            "default": true,
            "description": "Allows users to subscribe to feeds of search results",
            "factory": "Yes/No",
            "title": "Search RSS enabled",
            "type": "boolean"
        }
    },
    "value": true
},
{
    "name": "Products.CMFPlone.interfaces.syndication.ISiteSyndicationSettings.show_author_info",
    "schema": {
        "properties": {
            "default": true,
            "description": "Should feeds include author information",
            "factory": "Yes/No",
            "title": "Show author info",
            "type": "boolean"
        }
    },
    "value": true
},
{
    "name": "Products.CMFPlone.interfaces.syndication.ISiteSyndicationSettings.show_syndication_button",
    "schema": {
        "properties": {
            "description": "Makes it possible to customize syndication settings for particular folders and collections",
            "factory": "Yes/No",
            "title": "Show settings button",
            "type": "boolean"
        }
    },
    "value": null
}
```

(continues on next page)
```json
{
    "name": "Products.CMFPlone.interfaces.syndication.ISiteSyndicationSettings.show_syndication_link",
    "schema": {
        "properties": {
            "description": "Enable RSS link document action on the syndication content item."
        },
        "factory": "Yes/No",
        "title": "Show feed link",
        "type": "boolean"
    }
},
"value": null,
{
    "name": "Products.CMFPlone.interfaces.syndication.ISiteSyndicationSettings.site_rss_items",
    "schema": {
        "properties": {
            "additionalItems": true,
            "default": [
                "/news/aggregator"
            ],
            "description": "Paths to folders and collections to link to at the portal root."
        },
        "factory": "Tuple",
        "items": {
            "description": "",
            "factory": "Choice",
            "title": "",
            "type": "string",
            "vocabulary": {
                "@id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabularies.SyndicatableFeedItems"
            },
            "title": "Site RSS",
            "type": "array",
            "uniqueItems": true
        }
    },
    "value": [
        "/news/aggregator"
    ]
},
{
    "name": "plone.alignment_styles",
    "schema": {
        "properties": {
            "additionalItems": true,
            "default": [
                "Left|alignleft|alignleft",
                "Center|aligncenter|aligncenter",
                "Right|alignright|alignright",
                "Justify|alignjustify|alignjustify"
            ],
            "description": "Name|format|icon",
        }
    }
}
```
"factory": "List",
"items": {
  "description": "",
  "factory": "Text line (String)",
  "title": "",
  "type": "string"
},
"title": "Alignment styles",
"type": "array",
"uniqueItems": false
",
"value": [
  "Left|alignleft|alignleft",
  "Center|aligncenter|aligncenter",
  "Right|alignright|alignright",
  "Justify|alignjustify|alignjustify"
]
},
{
  "name": "plone.allow_anon_views_about",
  "schema": {
    "properties": {
      "default": false,
      "description": "If not selected only logged-in users will be able to view information about who created an item and when it was modified.",
      "factory": "Yes/No",
      "title": "Allow anyone to view 'about' information",
      "type": "boolean"
    }
  },
  "value": false
},
{
  "name": "plone.allow_external_login_sites",
  "schema": {
    "properties": {
      "additionalItems": true,
      "default": [],
      "description": "",
      "factory": "Tuple",
      "items": {
        "description": "",
        "factory": "Text line (String)",
        "title": "",
        "type": "string"
      },
      "title": "Allow external login sites",
      "type": "array",
      "uniqueItems": true
    }
  },
  "value": []
}
{
  "name": "plone.allowed_sizes",
  "schema": {
    "properties": {
      "default": [],
      "description": "",
      "factory": "Tuple",
      "items": {
        "description": "",
        "factory": "Text line (String)",
        "title": "",
        "type": "string"
      },
      "title": "Allowed sizes",
      "type": "array",
      "uniqueItems": true
    }
  },
  "value": []
}
"properties": {
  "additionalItems": true,
  "default": [
    "large 768:768",
    "preview 400:400",
    "mini 200:200",
    "thumb 128:128",
    "tile 64:64",
    "icon 32:32",
    "listing 16:16"
  ],
  "description": "Specify all allowed maximum image dimensions, one per line.
  The required format is &lt;name&gt; &lt;width&gt;&amp;:&lt;height&gt;",
  "factory": "List",
  "items": {
    "description": ",
    "factory": "Text line (String)",
    "title": ",
    "type": "string"
  },
  "title": "Allowed image sizes",
  "type": "array",
  "uniqueItems": false
},

"value": [
  "large 768:768",
  "preview 400:400",
  "mini 200:200",
  "thumb 128:128",
  "tile 64:64",
  "icon 32:32",
  "listing 16:16"
]
},

{name}: "plone.allowed_types",
schema": {
  "properties": {
    "additionalItems": true,
    "default": [
      "text/html",
      "text/x-web-textile"
    ],
    "description": "Select which formats are available for users as alternative to the default format. Note that if new formats are installed, they will be enabled for text fields by default unless explicitly turned off here or by the relevant installer.",
    "factory": "Tuple",
    "items": {
      "description": ",
      "factory": "Choice",
      "title": ",
      "type": "string",
      "vocabulary": {
        "@id": "http://localhost:55001/plone/@vocabularies/plone.app/vocabularies.AllowableContentTypes"
      }
    }
  }
}
"title": "Alternative formats",
"type": "array",
"uniqueItems": true
}
,
"value": [
  "text/html",
  "text/x-web-textile"
]
}
,
{name": "plone.always_show_selector",
"schema": {
  "properties": {
    "default": false,
    "description": ":",
    "factory": "Yes/No",
    "title": "Always show language selector",
    "type": "boolean"
  }
}
,
"value": false
}
,
{name": "plone.app.discussion.interfaces.IDiscussionSettings.anonymous_comments",
"schema": {
  "properties": {
    "default": false,
    "description": "If selected, anonymous users are able to post comments without logging in. It is highly recommended to use a captcha solution to prevent spam if this setting is enabled.",
    "factory": "Yes/No",
    "title": "Enable anonymous comments",
    "type": "boolean"
  }
}
,
"value": false
}
,
{name": "plone.app.discussion.interfaces.IDiscussionSettings.anonymous_email_enabled",
"schema": {
  "properties": {
    "default": false,
    "description": "If selected, anonymous user will have to give their email.",
    "factory": "Yes/No",
    "title": "Enable anonymous email field",
    "type": "boolean"
  }
}
,
"value": false
}
,
{name": "plone.app.discussion.interfaces.IDiscussionSettings.captcha",
"schema": {}
"schema": {  
  "properties": {  
    "default": "disabled",
    "description": "Use this setting to enable or disable Captcha validation for comments. Install plone.formwidget.captcha, plone.formwidget.recaptcha, collective.akismet, or collective.z3cform.norobots if there are no options available.",
    "factory": "Choice",
    "title": "Captcha",
    "type": "string",
    "vocabulary": {  
      @id": "http://localhost:55001/plone/@vocabularies/plone.app.discussion.vocabulary.CaptchaVocabulary"
    }
  }
  ,  
  "value": "disabled"  
  },  
  "name": "plone.app.discussion.interfaces.IDiscussionSettings.delete_own_comment_enabled",  
  "schema": {  
    "properties": {  
      "default": false,
      "description": "If selected, supports deleting of own comments for users with the \"Delete own comments\" permission.",
      "factory": "Yes/No",
      "title": "Enable deleting own comments",
      "type": "boolean"
    }
  }
  ,  
  "value": false  
  },  
  "name": "plone.app.discussion.interfaces.IDiscussionSettings.edit_comment_enabled",  
  "schema": {  
    "properties": {  
      "default": false,
      "description": "If selected, supports editing of comments for users with the \"Edit comments\" permission.",
      "factory": "Yes/No",
      "title": "Enable editing of comments",
      "type": "boolean"
    }
  }
  ,  
  "value": false  
  },  
  "items_total": 1779
}

17.3 Updating registry records

Updating an existing record: http
17.3. Updating registry records

```bash
PATCH /plone/@registry/ HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
   "plone.app.querystring.field.path.title": "Value"
}

curl


httpie

```

```bash

echo '{"plone.app.querystring.field.path.title": "Value"}' | http PATCH http://nohost/plone/@registry/ Accept:application/json Content-Type:application/json -a admin:secret
```
Types

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

Available content types in a Plone site can be listed and queried by accessing the /@types endpoint on any context (requires an authenticated user). The ‘addable’ key specifies if the content type can be added to the current context. The ‘layouts’ key specifies the defined views.

```
GET /plone/@types HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i http://nohost/plone/@types -H 'Accept: application/json' --user admin:secret
```

httpie

```
http http://nohost/plone/@types Accept:application/json -a admin:secret
```

python-requests

```
requests.get('http://nohost/plone/@types', headers={'Accept': 'application/json'},
              auth=('admin', 'secret'))
```

```
HTTP/1.1 200 OK
Content-Type: application/json

[  
  
```
]`

(continues on next page)
The API consumer can create, read, update, and delete a content-types schema.

<table>
<thead>
<tr>
<th>Verb</th>
<th>URL</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/@types/{type}</td>
<td>Add field/fieldset to content type schema</td>
</tr>
<tr>
<td>GET</td>
<td>/@types/{type}</td>
<td>Get the schema of a content type</td>
</tr>
<tr>
<td>PATCH</td>
<td>/@types/{type}</td>
<td>Update existing schema fields/fieldsets properties</td>
</tr>
<tr>
<td>PUT</td>
<td>/@types/{type}</td>
<td>Replace content-type schema</td>
</tr>
</tbody>
</table>

In addition to the above methods we can also do:
Verb | URL | Action
--- | --- | ---
GET | `/@type/{type}/{field/fieldset}` | Get field/fieldset properties
PATCH | `/@type/{type}/{field/fieldset}` | Update field/fieldset properties
DELETE | `/@type/{type}/{field/fieldset}` | Remove field/fieldset from schema

Note: Schema fields/fieldsets defined by behaviors are immutable and can NOT be changed via this RestAPI endpoint. See Dexterity Types controlpanel RestAPI endpoint for enabling/disabling behaviors.

### 18.1 Add schema fieldset/field with POST

To create a new fieldset, send a POST request to the `/@types/Document` endpoint. http

```bash
POST /plone/@types/Document HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
    "description": "Contact information",
    "factory": "fieldset",
    "title": "Contact Info"
}
```

curl

```bash
```

httpie

```bash
echo '{
    "description": "Contact information",
    "factory": "fieldset",
    "title": "Contact Info"
```

python-requests

```python
requests.post('http://nohost/plone/@types/Document', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'description': 'Contact information', 'factory': 'fieldset', 'title': 'Contact Info'}, auth=('admin', 'secret'))
```

Response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "behavior": "plone.dexterity.schema.generated",
```

(continues on next page)
To create a new field, send a POST request to the /@types/Document endpoint. http

```
POST /plone/@types/Document HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
  "description": "Email of the author",
  "factory": "Email",
  "required": true,
  "title": "Author email"
}
```

curl

-"Content-Type: application/json" --data-raw '{"description": "Email of the author",
  "factory": "Email", "required": true, "title": "Author email"}' --user admin:secret

httpie

echo '{
  "description": "Email of the author",
  "factory": "Email",
  "required": true,
  "title": "Author email"
}' | http POST http://nohost/plone/@types/Document Accept:application/json Content-
-Type:application/json -a admin:secret

python-requests

```
requests.post('http://nohost/plone/@types/Document', headers={'Accept': 'application/
  json', 'Content-Type': 'application/json'}, json={'description': 'Email of the
  author', 'factory': 'Email', 'required': True, 'title': 'Author email'}, auth=(
  'admin', 'secret'))
```

Response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
  "behavior": "plone.dexterity.schema.generated.plone_0_Document",
  "description": "Email of the author",
  "factory": "Email",
  "title": "Author email",
  "type": "string",
  "widget": "email"
}
```
For a complete list of available field types you can access `/@vocabularies/Fields` endpoint. 

```plaintext
GET /plone/@vocabularies/Fields HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```plaintext
curl -i http://nohost/plone/@vocabularies/Fields -H 'Accept: application/json' --user admin:secret
```

httpie

```plaintext
http http://nohost/plone/@vocabularies/Fields Accept:application/json -a admin:secret
```

python-requests

```python
requests.get('http://nohost/plone/@vocabularies/Fields', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "@id": "http://localhost:55001/plone/@vocabularies/Fields",
  "items": [
    {
      "title": "Choice",
      "token": "label_choice_field"
    },
    {
      "title": "Date",
      "token": "label_date_field"
    },
    {
      "title": "Date/Time",
      "token": "label_datetime_field"
    },
    {
      "title": "Email",
      "token": "Email"
    },
    {
      "title": "Email",
      "token": "label_email"
    },
    {
      "title": "File Upload",
      "token": "File Upload"
    },
    {
      "title": "Floating-point number",
      "token": "label_float_field"
    },
    {
      "title": "Full Name",
```

(continues on next page)
"token": "label_full_name"
},

{ "title": "Image",
  "token": "Image"
},

{ "title": "Integer",
  "token": "label_integer_field"
},

{ "title": "JSONField",
  "token": "JSONField"
},

{ "title": "Multiple Choice",
  "token": "label_multi_choice_field"
},

{ "title": "Password",
  "token": "label_password_field"
},

{ "title": "Relation Choice",
  "token": "Relation Choice"
},

{ "title": "Relation List",
  "token": "Relation List"
},

{ "title": "Rich Text",
  "token": "Rich Text"
},

{ "title": "Text",
  "token": "label_text_field"
},

{ "title": "Text line (String)",
  "token": "label_textline_field"
},

{ "title": "URL",
  "token": "URL"
},

{ "title": "Yes/No",
  "token": "label_boolean_field"
}

"items_total": 20}
18.2 Get the schema with GET

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

```
GET /plone/@types/Document HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl
```
```

httpie
```
```

python-requests
```
requests.get('http://nohost/plone/@types/Document', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

```
HTTP/1.1 200 OK
Content-Type: application/json+schema

{
  "fieldsets": [
    {
      "behavior": "plone",
      "fields": [
        "title",
        "description",
        "author_email",
        "text",
        "changeNote"
      ],
      "id": "default",
      "title": "Default"
    },
    {
      "behavior": "plone",
      "fields": [
        "allow_discussion",
        "exclude_from_nav",
        "id",
        "versioning_enabled",
        "table_of_contents"
      ],
      "id": "settings",
      "title": "Settings"
    },
    {
      "behavior": "plone",
      "fields": [
        "subjects",
        "language"
      ]
    }
  ]
}```

(continues on next page)
"relatedItems"
],
"id": "categorization",
"title": "Categorization"
},
{
"behavior": "plone",
"fields": [
  "effective",
  "expires"
],
"id": "dates",
"title": "Dates"
},
{
"behavior": "plone",
"fields": [
  "creators",
  "contributors",
  "rights"
],
"id": "ownership",
"title": "Ownership"
},
{
"behavior": "plone.dexterity.schema.generated",
"description": "Contact information",
"fields": [],
"id": "contact_info",
"title": "Contact Info"
},
"layouts": [
  "document_view"
],
"properties": {
"allow_discussion": {
  "behavior": "plone.allowdiscussion",
  "choices": [
    [True, Yes],
    [False, No]
  ],
  "description": "Allow discussion for this content object."
},
"enum": [True, False],
"enumNames": [Yes, No]
"factory": "Choice",
"title": "Allow discussion",
"type": "string",
"vocabulary": {
   "$id": "http://localhost:55001/plone/0sources/allow_discussion"
}
},
"author_email": {
   "behavior": "plone.dexterity.schema.generated.plone_0_Document",
   "description": "Email of the author",
   "factory": "Email",
   "title": "Author email",
   "type": "string",
   "widget": "email"
},
"changeNote": {
   "behavior": "plone.versioning",
   "description": "Enter a comment that describes the changes you made."
},
"contributors": {
   "behavior": "plone.dublincore",
   "description": "The names of people that have contributed to this item. Each contributor should be on a separate line."
},
"creators": {
   "behavior": "plone.dublincore",
   "description": "Persons responsible for creating the content of this item. Please enter a list of user names, one per line. The principal creator should come first."
}
"title": "Creators",
"type": "array",
"uniqueItems": true,
"widgetOptions": {
  "vocabulary": {
    "@id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabulary."Users"
  }
},
"description": {
  "behavior": "plone.dublincore",
  "description": "Used in item listings and search results."
},
"effective": {
  "behavior": "plone.dublincore",
  "description": "If this date is in the future, the content will not show up in listings and searches until this date."
},
"exclude_from_nav": {
  "behavior": "plone.excludefromnavigation",
  "default": false,
  "description": "If selected, this item will not appear in the navigation tree"
},
"expires": {
  "behavior": "plone.dublincore",
  "description": "When this date is reached, the content will no longer be visible in listings and searches."
},
"id": {
  "behavior": "plone.shortname",
  "description": "This name will be displayed in the URL."
},
"language": {
  "behavior": "plone.dublincore",
  "default": "en",
  "description": "",
  "factory": "Choice",
  "title": "Language"}
"type": "string",
"vocabulary": {
SupportedContentLanguages"
},
"relatedItems": {
  "additionalItems": true,
  "behavior": "plone.relateditems",
  "default": [],
  "description": "",
  "factory": "Relation List",
  "items": {
    "description": "",
    "factory": "Relation Choice",
    "title": "Related",
    "type": "string",
    "vocabulary": {
Catalog"
    }
  },
  "title": "Related Items",
  "type": "array",
  "uniqueItems": true,
  "widgetOptions": {
    "pattern_options": {
      "recentlyUsed": true
    },
    "vocabulary": {
Catalog"
    }
  }
},
"rights": {
  "behavior": "plone.dublincore",
  "description": "Copyright statement or other rights information on this item."
  "factory": "Text",
  "title": "Rights",
  "type": "string",
  "widget": "textarea"
},
"subjects": {
  "additionalItems": true,
  "behavior": "plone.dublincore",
  "description": "Tags are commonly used for ad-hoc organization of content."
  "factory": "Tuple",
  "items": {
    "description": "",
    "factory": "Text line (String)"
  },
  "title": "Tags",
  "type": "array",
  "uniqueItems": true,
The content type schema uses the JSON Schema format. The tagged values for the widgets are also exposed in the the "properties" attribute of the schema.

For Choice fields, their vocabulary or source will be linked to a vocabulary or querysource property (one or the other, never both):

- If a querysource property is included, that field is backed by an IQerysource. In that case, the source’s terms can’t be enumerated, and the terms need to be queried by issuing a request to the linked endpoint and including the user’s search terms in the ?query= parameter.
- If a vocabulary property is included, the field is backed by a vocabulary or another kind of iterable source.
The terms can then be enumerated by issuing a request to the linked endpoint.

See Vocabularies and Sources for details on these endpoints.

See Types Schema for a detailed documentation about the available field types.

To get one schema fieldset properties, access @types/Document/{fieldset} endpoint: http

```
GET /plone/@types/Document/contact_info HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl
```
```

httpie
```
```

python-requests
```
requests.get('http://nohost/plone/@types/Document/contact_info', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

HTTP/1.1 200 OK
Content-Type: application/json

```json
{
    "behavior": "plone.dexterity.schema.generated",
    "description": "Contact information",
    "fields": [],
    "id": "contact_info",
    "title": "Contact Info"
}
```

To get one schema field properties, access @types/Document/{field} endpoint: http

```
GET /plone/@types/Document/author_email HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl
```
```

httpie
```
```

python-requests
```
requests.get('http://nohost/plone/@types/Document/author_email', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```
18.3 Update schema with PATCH

To update content type schema defaults 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).

To update one or more schema field properties: http

```bash
PATCH /plone/@types/Document HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
    "properties": {
        "author_email": {
            "default": "foo@bar.com",
            "maxLength": 20,
            "minLength": 5
        }
    }
}
```

curl
curl -i -X PATCH http://nohost/plone/@types/Document -H 'Accept: application/json' -H 'Content-Type: application/json' --data-raw '{"properties": {"author_email": {"default": "foo@bar.com", "maxLength": 20, "minLength": 5}}}' --user admin:secret

httpie
echo '{
    "properties": {
        "author_email": {
            "default": "foo@bar.com",
            "maxLength": 20,
            "minLength": 5
        }
    }

python-requests
To change one or more fieldsets properties: http

requests.patch('http://nohost/plone/@types/Document', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'properties': {'author_email': {'default': 'foo@bar.com', 'maxLength': 20, 'minLength': 5}}, auth=('admin', 'secret'))

Response:

HTTP/1.1 204 No Content

18.3. Update schema with PATCH
To update one fieldset settings, we can also send a PATCH request to `@types/Document/{fieldset}` endpoint:

```plaintext
HTTP PATCH /plone/@types/Document/contact_info HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
    "description": "Contact information",
    "fields": [
        "author_email"
    ],
    "title": "Contact information"
}
```

curl


httpie

echo '{
    "description": "Contact information",
    "fields": ["author_email"],
    "title": "Contact information"

python-requests

```python
requests.patch('http://nohost/plone/@types/Document/contact_info', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'description': 'Contact information', 'fields': ['author_email'], 'title': 'Contact information'}, auth=('admin', 'secret'))
```

Response:

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

To update one field settings, we can also send a PATCH request to `@types/Document/{field}` endpoint:

```plaintext
HTTP PATCH /plone/@types/Document/author_email HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
    "description": "The e-mail address of the author",
    "maxLength": 20,
    "title": "Author's e-mail"
}
```

(continues on next page)
"minLength": 10,
"required": true,
"title": "Author e-mail"
}

```bash
curl -i -X PATCH http://nohost/plone/@types/Document/author_email -H 'Accept: application/json' -H 'Content-Type: application/json' --data-raw '{"description": "The e-mail address of the author", "maxLength": 20, "minLength": 10, "required": true, "title": "Author e-mail"}' --user admin:secret
```

```bash
echo '{
  "description": "The e-mail address of the author",
  "maxLength": 20,
  "minLength": 10,
  "required": true,
  "title": "Author e-mail"
```

```python
requests.patch('http://nohost/plone/@types/Document/author_email', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'description': 'The e-mail address of the author', 'maxLength': 20, 'minLength': 10, 'required': True, 'title': 'Author e-mail'}, auth=('admin', 'secret'))
```

Response:

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

### 18.4 Update schema with PUT

Use PUT when more changes are needed in one call, like create new fields/fieldsets, move fields to fieldset, remove multiple fields, etc. http

```bash
PUT /plone/@types/Document HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
    "fieldsets": [
    {
        "fields": [
            "author_email",
            "author_url",
            "author_name"
        ],
        "id": "author",
        "title": "Contact the author"
    }
}
```

(continues on next page)
"fields": [],
"id": "contact_info",
"title": "Contact info"
],
"layouts": [
  "thumbnail_view",
  "table_view"
],
"properties": {
  "allow_discussion": {
    "behavior": "plone.allowdiscussion",
    "choices": [
      ["True", "Yes"],
      ["False", "No"]
    ],
    "description": "Allow discussion for this content object.",
    "enum": ["True", "False"],
    "enumNames": ["Yes", "No"],
    "factory": "Choice",
    "title": "Allow discussion",
    "type": "string",
    "vocabulary": {
      "@id": "http://localhost:55001/plone/@sources/allow_discussion"
    }
  },
  "author_email": {
    "behavior": "plone.dexterity.schema.generated.plone_0_Document",
    "description": "Email of the author",
    "factory": "Email",
    "title": "Author email",
    "type": "string",
    "widget": "email"
  },
  "author_name": {
    "description": "Name of the author",
    "factory": "Text line (String)",
    "title": "Author name"
  },
  "author_url": {
    "description": "Author webpage",
    "factory": "URL",
    "maxLength": 30,
    "title": "Author url"
  }
}
"minLength": 5,
"title": "Author website"
},
"changeNote": {
"behavior": "plone.versioning",
"description": "Enter a comment that describes the changes you made."
"factory": "Text line (String)",
"title": "Change Note",
"type": "string"
},
"contributors": {
"additionalItems": true,
"behavior": "plone.dublincore",
"description": "The names of people that have contributed to this item. Each contributor should be on a separate line."
"factory": "Tuple",
"items": {
"description": "",
"factory": "Text line (String)",
"title": "",
"type": "string"
},
"title": "Contributors",
"type": "array",
"uniqueItems": true,
"widgetOptions": {
"vocabulary": {
"@id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabularies.Users"
}
}
},
"creators": {
"additionalItems": true,
"behavior": "plone.dublincore",
"description": "Persons responsible for creating the content of this item. Please enter a list of user names, one per line. The principal creator should come first."
"factory": "Tuple",
"items": {
"description": "",
"factory": "Text line (String)",
"title": "",
"type": "string"
},
"title": "Creators",
"type": "array",
"uniqueItems": true,
"widgetOptions": {
"vocabulary": {
"@id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabularies.Users"
}
}
},
"description": {  
"behavior": "plone.dublincore",
"description": "Enter a comment that describes the changes you made."
"factory": "Text line (String)",
"title": "Change Note",
"type": "string"
}
}
"description": "Used in item listings and search results."
"factory": "Text"
"title": "Summary"
"type": "string"
"widget": "textarea"
},
"effective": {
"behavior": "plone.dublincore"
"description": "If this date is in the future, the content will not show up in listings and searches until this date."
"factory": "Date/Time"
"title": "Publishing Date"
"type": "string"
"widget": "datetime"
},
"exclude_from_nav": {
"behavior": "plone.excludefromnavigation"
"description": "If selected, this item will not appear in the navigation tree"
"factory": "Yes/No"
"title": "Exclude from navigation"
"type": "boolean"
},
"expires": {
"behavior": "plone.dublincore"
"description": "When this date is reached, the content will no longer be visible in listings and searches."
"factory": "Date/Time"
"title": "Expiration Date"
"type": "string"
"widget": "datetime"
},
"id": {
"behavior": "plone.shortname"
"description": "This name will be displayed in the URL."
"factory": "Text line (String)"
"title": "Short name"
"type": "string"
},
"language": {
"behavior": "plone.dublincore"
"default": "en"
"description": ""
"factory": "Choice"
"title": "Language"
"type": "string"
"vocabulary": {
"@id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabularies.SupportedContentLanguages"
},
"relatedItems": {
"additionalItems": true
"behavior": "plone.relateditems"
"default": []
"description": ""
"factory": "Relation List",
"items": {
  "description": "",
  "factory": "Relation Choice",
  "title": "Related",
  "type": "string",
  "vocabulary": {
    →vocabularies.Catalog"
  },
  "title": "Related Items",
  "type": "array",
  "uniqueItems": true,
  "widgetOptions": {
    "pattern_options": {
      "recentlyUsed": true
    },
    "vocabulary": {
    →vocabularies.Catalog"
    }
  }
},

"rights": {
  "behavior": "plone.dublincore",
  "description": "Copyright statement or other rights information on this
  →item."
},

"factory": "Text",
"title": "Rights",
"type": "string",
"widget": "textarea"
},

"subjects": {
  "additionalItems": true,
  "behavior": "plone.dublincore",
  "description": "Tags are commonly used for ad-hoc organization of content.
  →",
  "factory": "Tuple",
  "items": {
    "description": "",
    "factory": "Text line (String)"
  },
  "title": "Tags",
  "type": "array",
  "uniqueItems": true,
  "widgetOptions": {
    "vocabulary": {
    →vocabularies.Keywords"
    }
  }
},

"table_of_contents": {
  "behavior": "plone.tableofcontents",
  "items": {
    "description": "",
    "factory": "Relation List",
    "title": "Related",
    "type": "string",
    "vocabulary": {
    →vocabularies.Catalog"
    }
  },
  "items": {
    "description": "",
    "factory": "Relation Choice",
    "title": "Related",
    "type": "string",
    "vocabulary": {
    →vocabularies.Catalog"
    }
  },
  "title": "Related Items",
  "type": "array",
  "uniqueItems": true,
  "widgetOptions": {
    "pattern_options": {
      "recentlyUsed": true
    },
    "vocabulary": {
    →vocabularies.Catalog"
    }
  }
},

"rights": {
  "behavior": "plone.dublincore",
  "description": "Copyright statement or other rights information on this
  →item."
},

"factory": "Text",
"title": "Rights",
"type": "string",
"widget": "textarea"
},

"subjects": {
  "additionalItems": true,
  "behavior": "plone.dublincore",
  "description": "Tags are commonly used for ad-hoc organization of content.
  →",
  "factory": "Tuple",
  "items": {
    "description": "",
    "factory": "Text line (String)"
  },
  "title": "Tags",
  "type": "array",
  "uniqueItems": true,
  "widgetOptions": {
    "vocabulary": {
    →vocabularies.Keywords"
    }
  }
},

"table_of_contents": {
  "behavior": "plone.tableofcontents",
  "items": {
    "description": "",
    "factory": "Relation List",
    "title": "Related",
    "type": "string",
    "vocabulary": {
    →vocabularies.Catalog"
    }
  },
  "items": {
    "description": "",
    "factory": "Relation Choice",
    "title": "Related",
    "type": "string",
    "vocabulary": {
    →vocabularies.Catalog"
    }
  },
  "title": "Related Items",
  "type": "array",
  "uniqueItems": true,
  "widgetOptions": {
    "pattern_options": {
      "recentlyUsed": true
    },
    "vocabulary": {
    →vocabularies.Catalog"
    }
  }
},

"rights": {
  "behavior": "plone.dublincore",
  "description": "Copyright statement or other rights information on this
  →item."
},

"factory": "Text",
"title": "Rights",
"type": "string",
"widget": "textarea"
},

"subjects": {
  "additionalItems": true,
  "behavior": "plone.dublincore",
  "description": "Tags are commonly used for ad-hoc organization of content.
  →",
  "factory": "Tuple",
  "items": {
    "description": "",
    "factory": "Text line (String)"
  },
  "title": "Tags",
  "type": "array",
  "uniqueItems": true,
  "widgetOptions": {
    "vocabulary": {
    →vocabularies.Keywords"
    }
  }
},

"table_of_contents": {
  "behavior": "plone.tableofcontents",
  "items": {
    "description": "",
    "factory": "Relation List",
    "title": "Related",
    "type": "string",
    "vocabulary": {
    →vocabularies.Catalog"
    }
  },
  "items": {
    "description": "",
    "factory": "Relation Choice",
    "title": "Related",
    "type": "string",
    "vocabulary": {
    →vocabularies.Catalog"
    }
  },
  "title": "Related Items",
  "type": "array",
  "uniqueItems": true,
  "widgetOptions": {
    "pattern_options": {
      "recentlyUsed": true
    },
    "vocabulary": {
    →vocabularies.Catalog"
    }
  }
},

"rights": {
  "behavior": "plone.dublincore",
  "description": "Copyright statement or other rights information on this
  →item."
},

"factory": "Text",
"title": "Rights",
"type": "string",
"widget": "textarea"}
"description": "If selected, this will show a table of contents at the top of the page."
},
"factory": "Yes/No",
"title": "Table of contents",
"type": "boolean"
}
},
"text": {
"behavior": "plone.richtext",
"description": "",
"factory": "Rich Text",
"title": "Text",
"type": "string",
"widget": "richtext"
}
},
"title": {
"behavior": "plone.dublincore",
"description": "",
"factory": "Text line (String)",
"title": "Title",
"type": "string"
}
},
"versioning_enabled": {
"behavior": "plone.versioning",
"default": true,
"description": "Enable/disable versioning for this document.",
"factory": "Yes/No",
"title": "Versioning enabled",
"type": "boolean"
}
},
"required": [
"title",
"author_email"
],
"title": "Page",
"type": "object"
}
httpie

```json
echo '{
  "fieldsets": [
    {
      "fields": [
        "author_email",
        "author_url",
        "author_name"
      ],
      "id": "author",
      "title": "Contact the author"
    },
    {
      "fields": [],
      "id": "contact_info",
      "title": "Contact info"
    }
  ],
  "layouts": [
    "thumbnail_view",
    "table_view"
  ],
  "properties": {
    "allow_discussion": {
      "behavior": "plone.allowdiscussion",
      "choices": [
        ["True", "Yes"],
        ["False", "No"
      ],
      "description": "Allow discussion for this content object.",
      "enum": ["True", "False"],
      "enumNames": ["Yes", "No"],
      "factory": "Choice",
      "title": "Allow discussion",
      "type": "string",
      "vocabulary": {
        "@id": "http://localhost:55001/plone/@sources/allow_discussion"
      }
    },
    "author_email": {
      "behavior": "plone.dexterity.schema.generated.plone_0_Document",
      "description": "Email of the author",
      "format": "email",
      "title": "Email of the author"
    }
  }
}'
```

(continues on next page)
"factory": "Email",
"title": "Author email",
"type": "string",
"widget": "email"
},
"author_name": {
"description": "Name of the author",
"factory": "Text line (String)",
"title": "Author name"
},
"author_url": {
"description": "Author webpage",
"factory": "URL",
"maxLength": 30,
"minLength": 5,
"title": "Author website"
},
"changeNote": {
"behavior": "plone.versioning",
"description": "Enter a comment that describes the changes you made.",
"factory": "Text line (String)",
"title": "Change Note",
"type": "string"
},
"contributors": {
"additionalItems": true,
"behavior": "plone.dublincore",
"description": "The names of people that have contributed to this item. Each contributor should be on a separate line.",
"factory": "Tuple",
"items": {
"description": "",
"factory": "Text line (String)",
"title": "",
"type": "string"
},
"title": "Contributors",
"type": "array",
"uniqueItems": true,
"widgetOptions": {
"vocabulary": {
"@id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabularies.Users"
}
}
},
"creators": {
"additionalItems": true,
"behavior": "plone.dublincore",
"description": "Persons responsible for creating the content of this item. Please enter a list of user names, one per line. The principal creator should come first.",
"factory": "Tuple",
"items": {
"description": "",
"factory": "Text line (String)",
"title": ""
}
}
18.4. Update schema with PUT
"factory": "Choice",
"title": "Language",
"type": "string",
"vocabulary": {
   "@id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabulariesSupportedContentLanguages"
}
},
"relatedItems": {
   "additionalItems": true,
   "behavior": "plone.relateditems",
   "default": [],
   "description": "",
   "factory": "Relation List",
   "items": {
      "description": "",
      "factory": "Relation Choice",
      "title": "Related",
      "type": "string",
      "vocabulary": {
         "@id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabularies.Catalog"
      }
   },
   "title": "Related Items",
   "type": "array",
   "uniqueItems": true,
   "widgetOptions": {
      "pattern_options": {
         "recentlyUsed": true
      },
      "vocabulary": {
         "@id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabularies.Catalog"
      }
   }
},
"rights": {
   "behavior": "plone.dublincore",
   "description": "Copyright statement or other rights information on this item."
},
"subjects": {
   "additionalItems": true,
   "behavior": "plone.dublincore",
   "description": "Tags are commonly used for ad-hoc organization of content."
},
"title": "Tags"
python-requests

requests.put('http://nohost/plone/@types/Document', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'fields': [{'id': 'author_email', 'author_url', 'author_name'}, 'id': 'author', 'title': 'Contact the author'], 'fields': [{'id': 'contact_info', 'title': 'Contact info'}], 'layouts': [{'thumbnail_view', 'table_view'], 'properties': {'allow_discussion': 'true', 'allow_discussion': 'false',

18.4. Update schema with PUT

$ python -m plone.app.testing.FunctionalTesting --keep --layer plone.testlayer.testing_layer

...
18.5 Removing schema field/fieldset with DELETE

Delete an existing schema field by sending a DELETE request to the URL of an existing schema field: http

```
DELETE /plone/@types/Document/author_email HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl
```
```

httpie
```
```

python-requests
```
requests.delete('http://nohost/plone/@types/Document/author_email', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

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

Delete an existing schema fieldset by sending a DELETE request to the URL of an existing schema fieldset: http

```
DELETE /plone/@types/Document/contact_info HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl
```
```

httpie
```
h http DELETE http://nohost/plone/@types/Document/contact_info Accept:application/json --a admin:secret
```

python-requests
```
requests.delete('http://nohost/plone/@types/Document/contact_info', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

Response:
```
HTTP/1.1 204 No Content
Content-Type: application/json
```
HTTP/1.1 204 No Content
CHAPTER 19

Types Schema

A detailed list of all available Zope Schema field types and their corresponding representation as JSON Schema.

19.1 TextLine

Zope Schema:

```python
zope.schema.TextLine(
    title=u'My field',
    description=u'My great field',
    default=u'foobar'
)
```

JSON Schema:

```json
{
    'type': 'string',
    'title': u'My field',
    'description': u'My great field',
    'default': u'foobar',
}
```

19.2 Text

Zope Schema:

```python
zope.schema.Text(
    title=u'My field',
    description=u'My great field',
    default=u'Lorem ipsum dolor sit amet',
)
```
19.3 Bool

Zope Schema:

```
zope.schema.Bool(
    title=u'My field',
    description=u'My great field',
    default=False,
)
```

JSON Schema:

```
{
    'type': 'boolean',
    'title': u'My field',
    'description': u'My great field',
    'default': False,
}
```

19.4 Float

Zope Schema:

```
zope.schema.Float(
    title=u'My field',
    description=u'My great field',
    min=0.0,
    max=1.0,
    default=0.5,
)
```

JSON Schema:

```
{
    'minimum': 0.0,
    'maximum': 1.0,
    'type': 'number',
}
```
19.5 Decimal

Zope Schema:

```python
zope.schema.Decimal(
    title=u'My field',
    description=u'My great field',
    min=Decimal(0),
    max=Decimal(1),
    default=Decimal(0.5),
)
```

JSON Schema:

```json
{
    'minimum': 0.0,
    'maximum': 1.0,
    'type': 'number',
    'title': u'My field',
    'description': u'My great field',
    'default': 0.5,
}
```

19.6 Int

Zope Schema:

```python
zope.schema.Int(
    title=u'My field',
    description=u'My great field',
    min=0,
    max=100,
    default=50,
)
```

JSON Schema:

```json
{
    'minimum': 0,
    'maximum': 100,
    'type': 'integer',
    'title': u'My field',
    'description': u'My great field',
    'default': 50,
}
```
19.7 Choice

Zope Schema:

```python
zope.schema.Choice(
    title=u'My field',
    description=u'My great field',
    vocabulary=self.dummy_vocabulary,
)
```

JSON Schema:

```json
{
    'type': 'string',
    'title': u'My field',
    'description': u'My great field',
    'enum': ['foo', 'bar'],
    'enumNames': ['Foo', 'Bar'],
    'choices': [('foo', 'Foo'), ('bar', 'Bar')],
}
```

19.8 List

Zope Schema:

```python
zope.schema.List(
    title=u'My field',
    description=u'My great field',
    min_length=1,
    value_type=schema.TextLine(
        title=u'Text',
        description=u'Text field',
        default=u'Default text',
    ),
    default=['foobar'],
)
```

JSON Schema:

```json
{
    'type': 'array',
    'title': u'My field',
    'description': u'My great field',
    'default': ['foobar'],
    'minItems': 1,
    'uniqueItems': False,
    'additionalItems': True,
    'items': {
        'type': 'string',
        'title': u'Text',
        'description': u'Text field',
        'default': u'Default text',
    },
}
```
19.9 Tuple

Zope Schema:

```python
field = zope.schema.Tuple(
    title=u'My field',
    value_type=schema.Int(),
    default=(1, 2),
)
```

JSON Schema:

```json
{
'type': 'array',
'title': u'My field',
'description': u'',
'uniqueItems': True,
'additionalItems': True,
'items': {
    'title': u'',
    'description': u'',
    'type': 'integer',
},
'default': (1, 2),
}
```

19.10 Set

Zope Schema:

```python
field = zope.schema.Set(
    title=u'My field',
    value_type=schema.TextLine(),
)
```

JSON Schema:

```json
{
'type': 'array',
'title': u'My field',
'description': u'',
'uniqueItems': True,
'additionalItems': True,
'items': {
    'title': u'',
    'description': u'',
    'type': 'string',
},
}
```

19.11 List of Choices

Zope Schema:
field = zope.schema.List(
    title=u'My field',
    value_type=zope.schema.Choice(
        vocabulary=self.dummy_vocabulary,
    ),
)

JSON Schema:

```json
{
    'type': 'array',
    'title': u'My field',
    'description': u'',
    'uniqueItems': True,
    'additionalItems': True,
    'items': {
        'title': u'',
        'description': u'',
        'type': 'string',
        'enum': ['foo', 'bar'],
        'enumNames': ['Foo', 'Bar'],
        'choices': [('foo', 'Foo'), ('bar', 'Bar')]
    }
}
```

### 19.12 Object

Zope Schema:

```python
zope.schema.Object(
    title=u'My field',
    description=u'My great field',
    schema=IDummySchema,
)
```

JSON Schema:

```json
{
    'type': 'object',
    'title': u'My field',
    'description': u'My great field',
    'properties': {
        'field1': {
            'title': u'Foo',
            'description': u'',
            'type': 'boolean'
        },
        'field2': {
            'title': u'Bar',
            'description': u'',
            'type': 'string'
        }
    }
}
```
19.13 RichText (plone.app.textfield)

Zope Schema:

```python
from plone.app.textfield import RichText
field = RichText(
    title=u'My field',
    description=u'My great field',
)
```

JSON Schema:

```json
{
    'type': 'string',
    'title': u'My field',
    'description': u'My great field',
    'widget': 'richtext',
}
```

19.14 Date

Zope Schema:

```python
zope.schema.Date(
    title=u'My field',
    description=u'My great field',
    default=date(2016, 1, 1),
)
```

JSON Schema:

```json
{
    'type': 'string',
    'title': u'My field',
    'description': u'My great field',
    'default': date(2016, 1, 1),
    'widget': u'date',
}
```

19.15 DateTime

Zope Schema:

```python
zope.schema.Datetime(
    title=u'My field',
    description=u'My great field',
)
```

JSON Schema:

```json
{
    'type': 'string',
    'title': u'My field',
    'description': u'My great field',
}
```
19.16 Email

Zope Schema:

```python
plone.schema.Email(
    title=u'My field',
    description=u'My great field',
)
```

JSON Schema:

```json
{
    'type': 'string',
    'title': u'My field',
    'description': u'My great field',
    'widget': u'email',
}
```

19.17 Password

Zope Schema:

```python
zope.schema.Password(
    title=u'My field',
    description=u'My great field',
)
```

JSON Schema:

```json
{
    'type': 'string',
    'title': u'My field',
    'description': u'My great field',
    'widget': u'password',
}
```

19.18 URI

Zope Schema:

```python
zope.schema.URI(
    title=u'My field',
)
```

(continues on next page)
description=u'My great field',
}

JSON Schema:
{
    'type': 'string',
    'title': u'My field',
    'description': u'My great field',
    'widget': u'url',
}
Available users in a Plone site can be created, queried, updated and deleted by interacting with the /@users endpoint on portal root (requires an authenticated user):

### 20.1 List Users

To retrieve a list of all current users in the portal, call the /@users endpoint with a GET request: http

```
GET /plone/@users HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i http://nohost/plone/@users -H 'Accept: application/json' --user admin:secret
```

httpie

```
http http://nohost/plone/@users Accept:application/json -a admin:secret
```

python-requests

```
requests.get('http://nohost/plone/@users', headers={'Accept': 'application/json'},
             auth=('admin', 'secret'))
```

The server will respond with a list of all users in the portal:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  
  "@id": "http://localhost:55001/plone/@users/admin",

(continues on next page)
```
This only works for Manager users, anonymous users or logged-in users without Manager rights are now allowed to list users. This is the example as an anonymous user: http

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

curl
```
curl -i http://nohost/plone/@users -H 'Accept: application/json'
```

httpie
```
http http://nohost/plone/@users Accept:application/json
```

python-requests
```
requests.get('http://nohost/plone/@users', headers={'Accept': 'application/json'})
```

The server will return a 401 Unauthorized status code

```
HTTP/1.1 401 Unauthorized
Content-Type: application/json
null
```

And this one as an user without the proper rights: http
The endpoint supports some basic filtering:

```http
GET /plone/@users?query=noa HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

```curl
```

```httpie
```

```python-requests
requests.get('http://nohost/plone/@users?query=noa', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

The server will respond with a list the filtered users in the portal with username starts with the query.

```http
HTTP/1.1 200 OK
Content-Type: application/json
```

```json
{
    "@id": "http://localhost:55001/plone/@users/noam",
    "description": "Professor of Linguistics",
    "email": "noam.chomsky@example.com",
    "fullname": "Noam Avram Chomsky",
    "home_page": "web.mit.edu/chomsky",
    "id": "noam",
    ...
}
```

(continues on next page)
"location": "Cambridge, MA",
"portrait": null,
"roles": [
  "Member"
],
"username": "noam"
}
]

The endpoint also takes a limit parameter that defaults to a maximum of 25 users at a time for performance reasons.

## 20.2 Create User

To create a new user, send a POST request to the global /@users endpoint with a JSON representation of the user you want to create in the body: http

```plaintext
POST /plone/@users HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
    "description": "Professor of Linguistics",
    "email": "noam.chomsky@example.com",
    "fullname": "Noam Avram Chomsky",
    "home_page": "web.mit.edu/chomsky",
    "location": "Cambridge, MA",
    "password": "colorlessgreenideas",
    "roles": [
        "Contributor"
    ],
    "username": "noamchomsky"
}

```

curl

```plaintext
curl -i -X POST http://nohost/plone/@users -H 'Accept: application/json' -H 'Content-Type: application/json' --data-raw '{"description": "Professor of Linguistics", "email": "noam.chomsky@example.com", "fullname": "Noam Avram Chomsky", "home_page": "web.mit.edu/chomsky", "location": "Cambridge, MA", "password": "colorlessgreenideas", "roles": ["Contributor"], "username": "noamchomsky"}' --user admin:secret
```

httpie

```plaintext
echo '{
    "description": "Professor of Linguistics",
    "email": "noam.chomsky@example.com",
    "fullname": "Noam Avram Chomsky",
    "home_page": "web.mit.edu/chomsky",
    "location": "Cambridge, MA",
    "password": "colorlessgreenideas",
    "roles": [
        "Contributor"
    ],
}
```
| "username": "noamchomsky"
|' | http POST http://nohost/plone/@users Accept:application/json Content-Type:application/json -a admin:secret

python-requests

requests.post('http://nohost/plone/@users', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'description': 'Professor of Linguistics', 'email': 'noam.chomsky@example.com', 'fullname': 'Noam Avram Chomsky', 'home_page': 'web.mit.edu/chomsky', 'location': 'Cambridge, MA', 'password': 'colorlessgreenideas', 'roles': ['Contributor'], 'username': 'noamchomsky'}, auth=('admin', 'secret'))

Note: By default, “username”, and “password” are required fields. If email login is enabled, “email” and “password” are required fields. All other fields in the example are optional.

The field “username” is not allowed when email login is enabled.

If the user has been created successfully, the server will respond with a status 201 (Created). The Location header contains the URL of the newly created user and the resource representation in the payload:

HTTP/1.1 201 Created
Content-Type: application/json
Location: http://localhost:55001/plone/@users/noamchomsky

{
   "@id": "http://localhost:55001/plone/@users/noamchomsky",
   "description": "Professor of Linguistics",
   "email": "noam.chomsky@example.com",
   "fullname": "Noam Avram Chomsky",
   "home_page": "web.mit.edu/chomsky",
   "id": "noamchomsky",
   "location": "Cambridge, MA",
   "portrait": null,
   "roles": [
      "Contributor"
   ],
   "username": "noamchomsky"
}

If no roles has been specified, then a default Member role is added as a sensible default.

20.3 Read User

To retrieve all details for a particular user, send a GET request to the /@users endpoint and append the user id to the URL: http

GET /plone/@users/noam HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0

curl
The server will respond with a 200 OK status code and the JSON representation of the user in the body:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "@id": "http://localhost:55001/plone/@users/noam",
  "description": "Professor of Linguistics",
  "email": "noam.chomsky@example.com",
  "fullname": "Noam Avram Chomsky",
  "home_page": "web.mit.edu/chomsky",
  "id": "noam",
  "location": "Cambridge, MA",
  "portrait": null,
  "roles": [
    "Member"
  ],
  "username": "noam"
}
```

The key 'roles' lists the globally defined roles for the user.

Only users with Manager rights are allowed to get other users’ information: http

```
GET /plone/@users/noam HTTP/1.1
Accept: application/json
Authorization: Basic bm9hbS1mYWtlOnNlY3JldA==
```

If the user lacks these rights, the server will respond with a 401 Unauthorized status code:
HTTP/1.1 401 Unauthorized
Content-Type: application/json
null

Also anonymous users are not allowed to get users’ information: http

GET /plone/@users/noam HTTP/1.1
Accept: application/json
curl

curl -i http://nohost/plone/@users/noam -H 'Accept: application/json'

httpie

http http://nohost/plone/@users/noam Accept:application/json

Python-Requests

requests.get('http://nohost/plone/@users/noam', headers={'Accept': 'application/json'})

If the user is an anonymous one, the server will respond with a 401 Unauthorized status code:

HTTP/1.1 401 Unauthorized
Content-Type: application/json
null

But each user is allowed to get its own information. http

GET /plone/@users/noam HTTP/1.1
Accept: application/json
Authorization: Basic bm9hbTpzZWNyZXQ=
curl

curl -i http://nohost/plone/@users/noam -H 'Accept: application/json' --user noam:secret

httpie

http http://nohost/plone/@users/noam Accept:application/json -a noam:secret

Python-Requests

requests.get('http://nohost/plone/@users/noam', headers={'Accept': 'application/json'}, auth=('noam', 'secret'))

In this case, the server will respond with a 200 OK status code and the JSON representation of the user in the body:

HTTP/1.1 200 OK
Content-Type: application/json

```json
{
    "id": "http://localhost:55001/plone/@users/noam",
}
```

(continues on next page)
"description": "Professor of Linguistics",
"email": "noam.chomsky@example.com",
"fullname": "Noam Avram Chomsky",
"home_page": "web.mit.edu/chomsky",
"id": "noam",
"location": "Cambridge, MA",
"portrait": null,
"roles": [
  "Member"
],
"username": "noam"
}

## 20.4 Update User

To update the settings of a user, send a PATCH request with the user details you want to amend to the URL of that particular user, e.g. if you want to update the email address of the admin user to: http

```
PATCH /plone/@users/noam HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
  "email": "avram.chomsky@example.com",
  "roles": {
    "Contributor": false
  }
}
```

curl

```
curl -i -X PATCH http://nohost/plone/@users/noam -H 'Accept: application/json' -H 'Content-Type: application/json' --data-raw '{"email": "avram.chomsky@example.com", "roles": {"Contributor": false}}' --user admin:secret
```

httpie

```
echo '{
  "email": "avram.chomsky@example.com",
  "roles": {
    "Contributor": false
  }
}' | http PATCH http://nohost/plone/@users/noam Accept:application/json Content-Type:application/json -a admin:secret
```

python-requests

```
requests.patch('http://nohost/plone/@users/noam', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'email': 'avram.chomsky@example.com', 'roles': {'Contributor': False}}, auth=('admin', 'secret'))
```

A successful response to a PATCH request will be indicated by a 204 No Content response:
Note: The ‘roles’ object is a mapping of a role and a boolean indicating adding or removing.

Any user is able to update their own properties and password (if allowed) by using the same request. The user portrait/avatar can also be updated using the same serialization as the file one: http

```
PATCH /plone/@users/noam HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
    "portrait": {
        "content-type": "image/gif",
        "data": "R0lGODlhAQABAIAAAP///wAAACwAAAAAAQABAAACAkQBADs=",
        "encoding": "base64",
        "filename": "image.gif"
    }
}
```

curl

curl -i -X PATCH http://nohost/plone/@users/noam -H 'Accept: application/json' -H 'Content-Type: application/json' --data-raw '{"portrait": {"content-type": "image/gif", "data": "R0lGODlhAQABAIAAAP///wAAACwAAAAAAQABAAACAkQBADs=", "encoding": "base64", "filename": "image.gif"}}' --user admin:secret

httpie

echo '{
    "portrait": {
        "content-type": "image/gif",
        "data": "R0lGODlhAQABAIAAAP///wAAACwAAAAAAQABAAACAkQBADs=",
        "encoding": "base64",
        "filename": "image.gif"
    }
}' | http PATCH http://nohost/plone/@users/noam Accept: application/json Content-Type: application/json -a admin:secret

grequests

```python
requests.patch('http://nohost/plone/@users/noam', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={"portrait": {"content-type": "image/gif", "data": "R0lGODlhAQABAIAAAP///wAAACwAAAAAAQABAAACAkQBADs=", "encoding": "base64", "filename": "image.gif"}}, auth=('admin', 'secret'))
```

A successful response to a PATCH request will be indicated by a 204 No Content response. Then asking for the user the portrait URL should be on the response:

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

(continues on next page)
Adding the portrait via the @user endpoint does not scale it since it’s assumed that the frontend will take care of it (resizing/cropping). If you still want that Plone to take care of the scaling using the default Plone behavior for portraits, you have to add the `scale` attribute to the request: http

```plaintext
PATCH /plone/@users/noam HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json


```

```

```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
20.5 Delete User

To delete a user send a DELETE request to the `/@users` endpoint and append the user id of the user you want to delete, e.g. to delete the user with the id johndoe: http

```
DELETE /plone/@users/noam HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl
```
curl -i -X DELETE http://nohost/plone/@users/noam -H 'Accept: application/json' --user admin:secret
```

httpie
```
http DELETE http://nohost/plone/@users/noam Accept:application/json -a admin:secret
```

python-requests
```
requests.delete('http://nohost/plone/@users/noam', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

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

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

20.6 User registration

Plone allows you to enable the auto registration of users. If it is enabled, then an anonymous user can register a new user using the user creation endpoint. This new user will have the role `Member` by default as the Plone registration process also does.

To create a new user send a POST request to the `@users` endpoint: http

```
POST /plone/@users HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
    "description": "Professor of Linguistics",
    "email": "noam.chomsky@example.com",
    "fullname": "Noam Avram Chomsky",
    "home_page": "web.mit.edu/chomsky",
    "location": "Cambridge, MA",
    "sendPasswordReset": true,
    "username": "noamchomsky"
}
```

curl
If the user should receive an email to set her password, you should pass `sendPasswordReset": true` in the JSON body of the request. Keep in mind that Plone will send a URL that points to the URL of the Plone site, which might just be your API endpoint.

If the user has been created, the server will respond with a 201 Created response:

```plaintext
HTTP/1.1 201 Created
Content-Type: application/json
Location: http://localhost:55001/plone/@users/noamchomsky

{
    "@id": "http://localhost:55001/plone/@users/noamchomsky",
    "description": "Professor of Linguistics",
    "email": "noam.chomsky@example.com",
    "fullname": "Noam Avram Chomsky",
    "home_page": "web.mit.edu/chomsky",
    "location": "Cambridge, MA",
    "id": "noamchomsky",
    "location": "Cambridge, MA",
    "portrait": null,
    "roles": [
       "Member"
    ],
    "username": "noamchomsky"
}
```

## 20.7 Reset User Password

Plone allows to reset a password for a user by sending a POST request to the user resource and appending `/reset-password` to the URL:
POST /plone/@users/noam/reset-password HTTP/1.1
Host: localhost:8080
Accept: application/json

The server will respond with a **200 OK** response and send an email to the user to reset her password.

The token that is part of the reset url in the email can be used to authorize setting a new password:

```plaintext
POST /plone/@users/noam/rest-password HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjdjMyMjI=
Content-Type: application/json

{"reset_token": "ef3d2aabacdc2345df63d6acf2edbef4", "new_password": "verysecret"}
```

```bash
```

```bash
httpie
```

```bash
echo '{
  "new_password": "verysecret",
  "reset_token": "ef3d2aabacdc2345df63d6acf2edbef4"
}' | http POST http://nohost/plone/@users/noam/rest-password Accept:application/json
```

```python
requests.post('http://nohost/plone/@users/noam/rest-password', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'new_password': 'verysecret', 'reset_token': 'ef3d2aabacdc2345df63d6acf2edbef4'}, auth=('admin', 'secret'))
```

### 20.7.1 Reset Own Password

Plone also allows a user to reset her own password directly without sending an email. The endpoint and the request is the same as above, but now the user can send the old password and the new password as payload:

```plaintext
POST /plone/@users/noam/reset-password HTTP/1.1
Host: localhost:8080
Accept: application/json
Content-Type: application/json

{
  'old_password': 'secret',
  'new_password': 'verysecret',
}

```

The server will respond with a **200 OK** response without sending an email.

To set the password with the old password you need either the **Set own password** or the **plone.app.controlpanel.UsersAndGroups** permission.
If an API consumer tries to send a password in the payload that is not the same as the currently logged in user, the server will respond with a **400 Bad Request** response.

### 20.7.2 Return Values

- **200 OK**
- **400 Bad Request**
- **403** (Unknown Token)
- **403** (Expired Token)
- **403** (Wrong user)
- **403** (Not allowed)
- **403** (Wrong password)
- **500 Internal Server Error** (server fault, can not recover internally)
Available groups in a Plone site can be created, queried, updated and deleted by interacting with the `/@groups` endpoint on portal root (requires an authenticated user):

### 21.1 List Groups

To retrieve a list of all current groups in the portal, call the `/@groups` endpoint with a GET request: http

```
GET /plone/@groups HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i http://nohost/plone/@groups -H 'Accept: application/json' --user admin:secret
```

httpie

```
http http://nohost/plone/@groups Accept:application/json -a admin:secret
```

python-requests

```
requests.get('http://nohost/plone/@groups', headers={'Accept': 'application/json'},
             auth=('admin', 'secret'))
```

The server will respond with a list of all groups in the portal:

```
HTTP/1.1 200 OK
Content-Type: application/json

[ ]
  |
  |
  "@id": "http://localhost:55001/plone/@groups/Administrators",
```

(continues on next page)
The endpoint supports some basic filtering: http
The server will respond with a list the filtered groups in the portal with groupname starts with the query.

The endpoint also takes a limit parameter that defaults to a maximum of 25 groups at a time for performance reasons.

HTTP/1.1 200 OK
Content-Type: application/json

[{
    "@id": "http://localhost:55001/plone/@groups/ploneteam",
    "description": "We are Plone",
    "email": "ploneteam@plone.org",
    "groupname": "ploneteam",
    "id": "ploneteam",
    "roles": [
        "Authenticated"
    ],
    "title": "Plone Team"
}]

### 21.2 Create Group

To create a new group, send a POST request to the global /@groups endpoint with a JSON representation of the group you want to create in the body: http

POST /plone/@groups HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{  
    "description": "The Plone Framework Team",
    "email": "fwt@plone.org",
    "groupname": "fwt",
    "groups": [  

(continues on next page)
"Administrators"
  },
  "roles": [
    "Manager"
  ],
  "title": "Framework Team",
  "users": [
    "admin",
    "test_user_1_"
  ]
}]

```bash
curl

curl -i -X POST http://nohost/plone/@groups -H 'Accept: application/json' -H 'Content-Type: application/json' --data-raw '["description": "The Plone Framework Team", "email": "fwt@plone.org", "groupname": "fwt", "groups": ["Administrators"], "roles": ["Manager"], "title": "Framework Team", "users": ["admin", "test_user_1_"]]' --user admin:secret
```
21.3 Read Group

To retrieve all details for a particular group, send a GET request to the `/@groups` endpoint and append the group id to the URL:

```
GET /plone/@groups/ploneteam HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

```
curl
```
curl -i http://nohost/plone/@groups/ploneteam -H 'Accept: application/json' --user admin:secret

```
httpie
```
```
http http://nohost/plone/@groups/ploneteam Accept:application/json -a admin:secret
```

```
python-requests
```
```
requests.get('http://nohost/plone/@groups/ploneteam', headers={"Accept": 'application/json"}, auth=('admin', 'secret'))
```

The server will respond with a 200 OK status code and the JSON representation of the group in the body:

```
HTTP/1.1 200 OK
Content-Type: application/json

|
"@id": "http://localhost:55001/plone/@groups/ploneteam",
```

(continues on next page)
"description": "We are Plone",
"email": "ploneteam@plone.org",
"groupname": "ploneteam",
"id": "ploneteam",
"roles": [
  "Authenticated"
],
"title": "Plone Team",
"users": {
  "@id": "http://localhost:5501/plone/@groups/ploneteam",
  "items": [],
  "items_total": 0
}
}

Batching is supported for the ‘users’ object.

## 21.4 Update Group

To update the settings of a group, send a PATCH request with the group details you want to amend to the URL of that particular group: http

```
PATCH /plone/@groups/ploneteam HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
  "email": "ploneteam2@plone.org",
  "users": {
    "test_user_1": false
  }
}
```

curl

curl -i -X PATCH http://nohost/plone/@groups/ploneteam -H 'Accept: application/json' -H 'Content-Type: application/json' --data-raw "{"email": "ploneteam2@plone.org", "users": {"test_user_1": false}}" --user admin:secret

httpie

```
echo '{
  "email": "ploneteam2@plone.org",
  "users": {
    "test_user_1": false
  }
}' | http PATCH http://nohost/plone/@groups/ploneteam Accept:application/json Content-_Type:application/json -a admin:secret
```

python-requests

```
requests.patch('http://nohost/plone/@groups/ploneteam', headers={'Accept':
  'application/json', 'Content-Type': 'application/json'}, json={'email':
  'ploneteam2@plone.org', 'users': {'test_user_1': False}}, auth=('admin', 'secret'))
```
Note: The ‘users’ object is a mapping of a user_id and a boolean indicating adding or removing from the group.

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

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

## 21.5 Delete Group

To delete a group send a DELETE request to the `/@groups` endpoint and append the group id of the group you want to delete:

```
DELETE /plone/@groups/ploneteam HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2Vjc2VjcmV0
```

**curl**

```
curl -i -X DELETE http://nohost/plone/@groups/ploneteam -H 'Accept: application/json' -u admin:secret
```

**httpie**

```
http DELETE http://nohost/plone/@groups/ploneteam Accept:application/json -a admin:secret
```

**python-requests**

```
requests.delete('http://nohost/plone/@groups/ploneteam', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

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

```
HTTP/1.1 204 No Content
```
This endpoint will search for all the available principals in the local PAS plugins given a query string. We call a principal to any user or group in the system (requires an authenticated user):

### 22.1 Search Principals

To retrieve a list of principals given a search string, call the `/@principals` endpoint with a GET request and a search query parameter:

```
GET /plone/@principals?search=ploneteam HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

**curl**

```
```

**httpie**

```
```

**python-requests**

```
requests.get('http://nohost/plone/@principals?search=ploneteam', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

The server will respond with a list of the users and groups in the portal that match the query string:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

(continues on next page)
```json
{
    "groups": [
        {
            "@id": "http://localhost:55001/plone/@groups/ploneteam",
            "description": "We are Plone",
            "email": "ploneteam@plone.org",
            "groupname": "ploneteam",
            "id": "ploneteam",
            "roles": ["Authenticated"],
            "title": "Plone Team"
        }
    ],
    "users": []
}
```
Available roles in a Plone site can be queried by interacting with the /@roles endpoint on portal root (requires an authenticated user):

### 23.1 List Roles

To retrieve a list of all roles in the portal, call the /@roles endpoint with a GET request: http

```bash
GET /plone/@roles HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i http://nohost/plone/@roles -H 'Accept: application/json' --user admin:secret
```

httpie

```
http http://nohost/plone/@roles Accept:application/json -a admin:secret
```

python-requests

```
requests.get('http://nohost/plone/@roles', headers={'Accept': 'application/json'},
               auth=('admin', 'secret'))
```

The server will respond with a list of all roles in the portal:

```json
HTTP/1.1 200 OK
Content-Type: application/json

[  
  {  
    "@id": "http://localhost:55001/plone/@roles/Contributor",
    "id": "Contributor",
    "title": "Contributor",
    "description": "A user who can contribute to the site.
```
The role title is the translated role title as displayed in Plone’s “Users and Groups” control panel.
CHAPTER 24

Components

Warning: The @components endpoint is deprecated and has been removed in plone.restapi 1.0b1. Breadcrumbs and Navigation are now top-level endpoints.

How to get pages components (i.e. everything but the main content), like breadcrumbs, navigations, actions, etc.
CHAPTER 25

Breadcrumbs

Getting the breadcrumbs for the current page: http

```
GET /plone/front-page/@breadcrumbs HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i http://nohost/plone/front-page/@breadcrumbs -H 'Accept: application/json' --
˓→user admin:secret
```

httpie

```
http http://nohost/plone/front-page/@breadcrumbs Accept:application/json -a
˓→admin:secret
```

python-requests

```
requests.get('http://nohost/plone/front-page/@breadcrumbs', headers={'Accept':
˓→'application/json'}, auth=('admin', 'secret'))
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

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

(continues on next page)
"root": "http://localhost:55001/plone"
26.1 Top-Level Navigation

Getting the top navigation items: http

```
GET /plone/front-page/@navigation HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i http://nohost/plone/front-page/@navigation -H 'Accept: application/json' --user admin:secret
```

httpie

```
http http://nohost/plone/front-page/@navigation Accept:application/json -a admin:secret
```

python-requests

```
requests.get('http://nohost/plone/front-page/@navigation', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "@id": "http://localhost:55001/plone/front-page/@navigation",
   "items": [
      {
         "@id": "http://localhost:55001/plone",
      
```
"description": "", "items": [], "review_state": null, "title": "Home"},
{
  "@id": "http://localhost:55001/plone/front-page",
  "description": "Congratulations! You have successfully installed Plone.",
  "items": [],
  "review_state": "private",
  "title": "Welcome to Plone"
}
]

26.2 Navigation Tree

Getting the navigation item tree providing a expand.navigation.depth parameter: http

<table>
<thead>
<tr>
<th>Method</th>
<th>URL</th>
<th>Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/plone/front-page/@navigation?expand.navigation.depth=4</td>
<td>Accept: application/json, Authorization: Basic YWRtaW46c2VjcmV0</td>
</tr>
</tbody>
</table>

curl


httpie

http 'http://nohost/plone/front-page/@navigation?expand.navigation.depth=4'
   -a admin:secret

python-requests

requests.get('http://nohost/plone/front-page/@navigation?expand.navigation.depth=4',
               headers={'Accept': 'application/json'}, auth=('admin', 'secret'))

Example response:

HTTP/1.1 200 OK
Content-Type: application/json

{　...
"@id": "http://localhost:55001/plone/front-page",
"description": "Congratulations! You have successfully installed Plone."

"items": [],
"review_state": "private",
"title": "Welcome to Plone"

"@id": "http://localhost:55001/plone/folder",
"description": "",
"items": [

  "@id": "http://localhost:55001/plone/folder/subfolder1",
  "description": "",
  "items": [

    "@id": "http://localhost:55001/plone/folder/subfolder1/thirdlevelfolder",
    "description": "",
    "items": [

      "@id": "http://localhost:55001/plone/folder/subfolder1/thirdlevelfolder/fourthlevelfolder",
      "description": "",
      "items": [],
      "review_state": "private",
      "title": "Fourth Level Folder",
      "use_view_action_in_listings": false
    ]
  ]
]

"review_state": "private",
"title": "SubFolder 1",
"use_view_action_in_listings": false

"@id": "http://localhost:55001/plone/folder/subfolder2",
"description": "",
"items": [],
"review_state": "private",
"title": "SubFolder 2",
"use_view_action_in_listings": false

"@id": "http://localhost:55001/plone/folder/doc1",
"description": "",
"items": [],
"review_state": "private",
"title": "A document",
"use_view_action_in_listings": false

"review_state": "private",
"title": "Some Folder"
```
},
{
    "@id": "http://localhost:55001/plone/folder2",
    "description": 
    "items": [],
    "review_state": "private",
    "title": "Some Folder 2"
}
```
27.1 Top-Level Navigation

Getting the top navigation items: http

```
GET /plone/folder/@contextnavigation HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i http://nohost/plone/folder/@contextnavigation -H 'Accept: application/json' --user admin:secret
```

httpie

```
```

python-requests

```
requests.get('http://nohost/plone/folder/@contextnavigation', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "@id": "http://localhost:55001/plone/folder/@contextnavigation",
  "available": true,
  "has_custom_name": false,
  "items": [ (continues on next page) ]
```
The @contextnavigation endpoint uses the same semantics as the classic Plone navigation portlet, largely through reusing the same code. Instead of storing the “portlet” configuration in a portlet assignment storage, you can pass these as parameters to the service or expand component.

You can provide these parameters:

- **name** - The title of the navigation tree.
• **root_path** - Root node path, can be “frontend path”, derived from router

• **includeTop** - Bool, Include top nodeschema

• **currentFolderOnly** - Bool, Only show the contents of the current folder.

• **topLevel** - Int, Start level

• **bottomLevel** - Int, Navigation tree depth

• **no_icons** - Bool, Suppress Icons

• **thumb_scale** - String, Override thumb scale

• **no_thumbs** - Bool, Suppress thumbs

You should prefix these parameters with `expand.contextnavigation`, so a request would look like:

http://localhost:55001/plone/?expand.contextnavigation.topLevel=1&expand.contextnavigation.name=Custom+name
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.

### 28.1 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.

### 28.2 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.

<table>
<thead>
<tr>
<th>Python</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>time(19, 45, 55)</td>
<td>'19:45:55'</td>
</tr>
<tr>
<td>date(2015, 11, 23)</td>
<td>'2015-11-23'</td>
</tr>
<tr>
<td>datetime(2015, 11, 23, 19, 45, 55)</td>
<td>'2015-11-23T19:45:55'</td>
</tr>
<tr>
<td>DateTime('2015/11/23 19:45:55')</td>
<td>'2015-11-23T19:45:55'</td>
</tr>
</tbody>
</table>

### 28.3 RichText fields

RichText fields will be serialized as follows:

A RichTextValue like
RichTextValue(u'<p>Hallochén</p>',
    mimeType='text/html',
    outputMimeType='text/html')

will be serialized to

```json
{
  "data": "<p>Hallochén</p>",
  "content-type": "text/html",
  "encoding": "utf-8"
}
```

## 28.4 File / Image Fields

### 28.4.1 Download (serialization)

For download, a file field will be serialized to a mapping that contains the file’s most basic metadata, and a hyperlink
that the client can follow to download the file:

```json
{
  "...": 
  ,
  "@type": "File",
  "title": "My file",
  "file": {
    "content-type": "application/pdf",
    "download": "http://localhost:55001/plone/file/@@download/file",
    "filename": "file.pdf",
    "size": 74429
  }
}
```

That URL in the `download` property points to the regular Plone download view. The client can send a `GET` request
to that URL with an `Accept` header containing the MIME type indicated in the `content-type` property, and will
get a response containing the file.

Image fields are serialized in the same way, except that their serialization contains their `width` and `height`, and an
additional property `scales` that contains a mapping with the available image scales. Image URLs are created using
the UID-based URL that changes each time the image is modified, so these URLs can be properly cached:

```json
{
  "icon": {
    "download": "http://localhost:55001/plone/image/@@images/8eed3f80-5e1f-4115-85b8-650a10a6ca84.png",
    "height": 32,
    "width": 24
  },
  "large": {
    "height": 768,
    "width": 576
  },
  "...": {}
}
```
28.4.2 Upload (deserialization)

For file or image fields, the client must provide the file’s data as a mapping containing the file data and some additional metadata:

- **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"
    }
}
```

28.5 Relations

28.5.1 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.

28.5.2 Deserialization

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

- **UID**
- **path**
- **URL**
- **intid**

Specify relations by UID:

```
{
    "relatedItems": [
        "158e5361282647e39bf0698fe238814b",
    ]
}
```
28.6 Next/Previous/Parent Navigation

The response body of a GET request contains three attributes that allows navigating to the parent and the next and previous sibling in the container the current document is located.

28.6.1 Parent

The “parent” attribute points to the parent container of the current content object.

```json
{
  "parent": {
    "@id": "http://nohost/plone/folder-with-items",
    "@type": "Folder",
    "title": "Folder with items",
    "description": "This is a folder with two documents",
  }
}
```

28.6.2 Previous Item

The “previous_item” attribute points to the sibling that is located before the current element in the parent container (Plone uses the getObjectPositionInParent attribute to sort content objects within a folderish container).

```json
{
  "previous_item": {
    "@id": "http://nohost/plone/folder-with-items/item-1",
    ...
  }
}
```
28.6.3 Next Item

The “next_item” attribute points to the sibling that is located after the current element in the parent container (Plone uses the getObjectPositionInParent attribute to sort content objects within a folderish container).
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 context (a *collection* in HTTP REST terms) 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: http

```
GET /plone/@search?sort_on=path HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i 'http://nohost/plone/@search?sort_on=path' -H 'Accept: application/json' --
    -u admin:secret
```

httpie

```
http 'http://nohost/plone/@search?sort_on=path' Accept:application/json -a
    --admin:secret
```

python-requests

```
requests.get('http://nohost/plone/@search?sort_on=path', headers={'Accept':
    'application/json'}, auth=('admin', 'secret'))
```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "@id": "http://localhost:55001/plone/@search",
    "items": [
        {
            "@id": "http://localhost:55001/plone/front-page",
            "@type": "Document",
            "description": "Congratulations! You have successfully installed Plone.",
            "review_state": "private",
            "title": "Welcome to Plone"
        }
    ],
    "items_total": 1
}

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.

**Note:** A search invoked on a container will by default include the container itself as part of the search results. This is the same behavior as displayed by `ZCatalog`, which is used internally. If you add the query string parameter `depth=1` to your search, you will only get immediate children of the container, and the container itself also won’t be part of the results. See the Plone docs on searching for content within a folder for more details.

**Note:** Search results will be batched if the size of the resultset exceeds the batch size. See `Batching` for more details on how to work with batched results.

**Warning:** The @@search view or the Plone LiveSearch widget are coded in a way that the SearchableText parameter is expanded by including a * wildcard at the end. This is done in order to match also the partial results of the beginning of a search term(s). The plone.restapi @@search endpoint will not do that for you. You’ll have to add it if you want to keep this feature.

### 29.1 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 HTTP/1.1
```

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.

### 29.1.1 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:

```python
cpy = {'path': {'query': '/folder1',
                   'depth': 2}}
```

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

```plaintext
http
GET /plone/@search?sort_on=path&path.query=%2Fplone%2Ffolder1&path.depth=1 HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0

```

```
  curl
  →depth=1' -H 'Accept: application/json' --user admin:secret

  httpie
  →depth=1' Accept:application/json -a admin:secret

  python-requests
  requests.get('http://nohost/plone/@search?sort_on=path&path.query=%2Fplone%2Ffolder1&
  →path.depth=1', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```  

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    →depth=1",
    "items": [
        {
            "@id": "http://localhost:55001/plone/folder1/folder2",
            "@type": "Folder",
            "description": "",
            "review_state": "private",
            "title": "Folder 2"
        }
    ],
    "items_total": 1
}
```

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

### 29.1.2 Restricting search to multiple paths

To restrict search to multiple paths, the original query as a Python dictionary would look like this (with an optional depth and sort_on):

```python
cpy = {'path': {'query': ('/folder', '/folder2'),
                   'depth': 2},
                   'sort_on': 'path'}
```

29.1. Query format
This dictionary will need to be flattened in dotted notation in order to pass it in a query string. In order to specify multiple paths, simply repeat the query string parameter (the `requests` module will do this automatically for you if you pass it a list of values for a query string parameter).

```
GET /plone/@search?sort_on=path&path.query=%2Fplone%2Ffolder1&path.query=%2Fplone
    →%2Ffolder2&path.depth=2 HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl
```
    →query=%2Fplone%2Ffolder2&path.depth=2' -H 'Accept: application/json' --user,
    →admin:secret
```

httpie
```
    →query=%2Fplone%2Ffolder2&path.depth=2' Accept:application/json -a admin:secret
```

```
python-requests
```
```
requests.get('http://nohost/plone/@search?sort_on=path&path.query=%2Fplone%2Ffolder1&
    →path.query=%2Fplone%2Ffolder2&path.depth=2', headers={'Accept': 'application/json'},
    →auth=('admin', 'secret'))
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        →query=%2Fplone%2Ffolder2&path.depth=2",
    "items": [
        {
            "@id": "http://localhost:55001/plone/folder1",
            "$type": "Folder",
            "description": "",
            "review_state": "private",
            "title": "Folder 1"
        },
        {
            "@id": "http://localhost:55001/plone/folder1/doc1",
            "$type": "Document",
            "description": "",
            "review_state": "private",
            "title": "Lorem Ipsum"
        },
        {
            "@id": "http://localhost:55001/plone/folder2",
            "$type": "Folder",
            "description": "",
            "review_state": "private",
            "title": "Folder 2"
        },
        {
            "@id": "http://localhost:55001/plone/folder2/doc2",
            "$type": "Document",

(continues on next page)
29.1.3 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 request. This means that for value 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:int=42 HTTP/1.1
Accept: application/json
```

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

29.2 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&metadata_fields=created HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl
```
```

httpie
```
```

python-requests
```
requests.get('http://nohost/plone/@search?SearchableText=lorem&metadata_fields=modified&metadata_fields=created', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

29.2. Retrieving additional metadata
HTTP/1.1 200 OK
Content-Type: application/json

{
  "@id": "http://localhost:55001/plone/@search?SearchableText=lorem&metadata_fields=modified&metadata_fields=created",
  "items": [
    {
      "@id": "http://localhost:55001/plone/doc1",
      "@type": "Document",
      "created": "1995-07-31T13:45:00",
      "description": "",
      "modified": "1995-07-31T17:30:00",
      "review_state": "private",
      "title": "Lorem Ipsum"
    }
  ],
  "items_total": 1
}

The metadata from those columns will then 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.

Note: There is a difference between the full set of fields contained in an object and the set of all possible metadata columns that can be specified with metadata_fields. In other words, using metadata_fields=_all will produce objects with a set of fields that is generally smaller than the set of fields produced by fullobjects (see next section). Briefly, the fields in metadata_fields=_all are a subset of fullobjects. A consequence of this is that certain fields can not be specified with metadata_fields. Doing so will result in a TypeError "No converter for making <...> JSON compatible." In ZCatalog terms, this reflects the difference between catalog brains and objects that have been woken up.

29.3 Retrieving full objects

If the data provided as metadata is not enough, you can retrieve search results as full serialized objects equivalent to what the resource GET request would produce.

You do so by specifying the fullobjects parameter: http

```plaintext
GET /plone/@search?SearchableText=lorem&fullobjects=1 HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
```

httpie

```
http 'http://nohost/plone/@search?SearchableText=lorem&fullobjects=1' -a admin:secret
```
```python
requests.get('http://nohost/plone/@search?SearchableText=lorem&fullobjects=1',
       headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "@id": "http://localhost:55001/plone/@search?SearchableText=lorem&fullobjects=1",
    "items": [
        {
            "@components": {
                "actions": {
                    "@id": "http://localhost:55001/plone/doc1/@actions"
                },
                "breadcrumbs": {
                    "@id": "http://localhost:55001/plone/doc1/@breadcrumbs"
                },
                "contextnavigation": {
                    "@id": "http://localhost:55001/plone/doc1/@contextnavigation"
                },
                "navigation": {
                    "@id": "http://localhost:55001/plone/doc1/@navigation"
                },
                "types": {
                    "@id": "http://localhost:55001/plone/doc1/@types"
                },
                "workflow": {
                    "@id": "http://localhost:55001/plone/doc1/@workflow"
                }
            },
            "@id": "http://localhost:55001/plone/doc1",
            "@type": "Document",
            "UID": "SomeUUID000000000000000000000002",
            "allow_discussion": false,
            "changeNote": "",
            "contributors": [],
            "created": "1995-07-31T13:45:00",
            "creators": [
                "test_user_1_"
            ],
            "description": "",
            "effective": null,
            "exclude_from_nav": false,
            "expires": null,
            "id": "doc1",
            "is_folderish": false,
            "layout": "document_view",
            "modified": "1995-07-31T17:30:00",
            "next_item": [],
            "parent": {
                "@id": "http://localhost:55001/plone",
                "@type": "Plone Site",
                "description": "",
                "title": "Plone site"
            }
        }
    ]
}
```
"previous_item": {
  "@id": "http://localhost:55001/plone/front-page",
  "@type": "Document",
  "description": "Congratulations! You have successfully installed Plone.",
  "title": "Welcome to Plone"
},
"relatedItems": [],
"review_state": "private",
"rights": "",
"subjects": [],
"table_of_contents": null,
"text": null,
"title": "Lorem Ipsum",
"version": "current",
"versioning_enabled": true,
"working_copy": null,
"working_copy_of": null
},
"items_total": 1
}

**Warning:** Be aware that this might induce performance issues when retrieving a lot of resources. Normally the search just serializes catalog brains, but with `fullobjects`, we wake up all the returned objects.

### 29.4 Restrict search results to Plone’s search settings

By default the search endpoint is not excluding any types from its results. To allow the search to follow Plone’s search settings schema, pass the `use_site_search_settings=1` to the `@search` endpoint request. By doing this, the search results will be filtered based on the defined types to be searched and will be sorted according to the default sorting order.
TUS resumable upload

plone.restapi supports the TUS Open Protocol for resumable file uploads. There is a `@tus-upload` endpoint to upload a file and a `@tus-replace` endpoint to replace an existing file.

### 30.1 Creating an Upload URL

**Note:** POST requests to the `@tus-upload` endpoint are allowed on all IFolderish content types (e.g. Folder).

To create a new upload, send a POST request to the `@tus-upload` endpoint. http

```bash
POST /plone/folder/@tus-upload HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Tus-Resumable: 1.0.0
Upload-Length: 8
Upload-Metadata: filename dGVzdC50eHQ=,content-type dGV4dC9wbGFpbw==
```

curl

```bash
```

httpie

```bash
```

python-requests

```bash
```
requests.post('http://nohost/plone/folder/@tus-upload', headers={'Accept': 'application/json', 'Tus-Resumable': '1.0.0', 'Upload-Length': '8', 'Upload-Metadata': 'filename dGVzdC50eHQ=,content-type dGV4dC9wbGFpbw=='}, auth=('admin', 'secret'))

The server will return a temporary upload URL in the location header of the response:

```
HTTP/1.1 201 Created
Location: http://localhost:55001/plone/folder/@tus-upload/032803b64ad746b3ab46d9223ea3d90f
Tus-Resumable: 1.0.0
```

The file can then be uploaded in the next step to that temporary URL.

### 30.2 Uploading a File

**Note:** PATCH requests to the `@tus-upload` endpoint are allowed on all IContentish content types.

Once a temporary upload URL has been created, a client can send a PATCH request to upload a file. The file content should be sent in the body of the request:

```
PATCH /plone/folder/@tus-upload/032803b64ad746b3ab46d9223ea3d90f HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Tus-Resumable: 1.0.0
Upload-Offset: 3
Content-Type: application/offset+octet-stream
defgh
```

When just a single file is uploaded at once, the server will respond with a 204: No Content response after a successful upload. The HTTP location header contains the URL of the newly created content object:

```
HTTP/1.1 204 No Content
Location: http://localhost:55001/plone/folder/document-2016-10-21
Tus-Resumable: 1.0.0
Upload-Offset: 8
```

### 30.3 Partial Upload

TUS allows partial upload of files. A partial file is also uploaded by sending a PATCH request to the temporary URL:

```
PATCH /plone/folder/@tus-upload/032803b64ad746b3ab46d9223ea3d90f HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Tus-Resumable: 1.0.0
Upload-Offset: 0
Content-Type: application/offset+octet-stream
abc
defgh
```

224 Chapter 30. TUS resumable upload
The server will also respond with a **204: No content** response. Though, instead of providing the final file URL in the ‘location’ header, the server provides an updated ‘Upload-Offset’ value, to tell the client the new offset:

```
HTTP/1.1 204 No Content
Tus-Resumable: 1.0.0
Upload-Offset: 3
```

When the last partial file has been uploaded, the server will contain the final file URL in the ‘location’ header.

### 30.4 Replacing Existing Files

TUS can also be used to replace an existing file by sending a POST request to the `@tus-replace` endpoint instead.

```
POST /plone/myfile/@tus-replace HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Tus-Resumable: 1.0.0
Upload-Length: 8
Upload-Metadata: filename dGVzdC50eHQ=,content-type dGV4dC9wbGFpbg==
```

The server will respond with a **201: Created** status and return the URL of the temporary created upload resource in the location header of the response:

```
HTTP/1.1 201 Created
Location: http://localhost:55001/plone/folder/@tus-upload/
Tus-Resumable: 1.0.0
```

The file can then be uploaded to that URL using the PATCH method in the same way as creating a new file:

```
PATCH /plone/myfile/@tus-upload/4e465958b24a46ec8657e6f3be720991 HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Tus-Resumable: 1.0.0
Upload-Offset: 0
Content-Type: application/offset+octet-stream
abcdefg
```

The server will respond with a **204: No Content** response and the final file URL in the HTTP location header:

```
HTTP/1.1 204 No Content
Location: http://localhost:55001/plone/myfile
Tus-Resumable: 1.0.0
Upload-Offset: 8
```

### 30.5 Asking for the Current File Offset

To ask the server for the current file offset, the client can send a HEAD request to the upload URL: `http`

```
HEAD /plone/folder/@tus-upload/032803b64ad746b3ab46d9223ea3d90f HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Tus-Resumable: 1.0.0
```
curl

```bash
curl -i -X HEAD http://nohost/plone/folder/@tus-upload/
  -032803b64ad746b3ab46d9223ea3d90f -H 'Accept: application/json' -H 'Tus-Resumable: 1.0.0' --user admin:secret
```

httpie

```bash
http HEAD http://nohost/plone/folder/@tus-upload/032803b64ad746b3ab46d9223ea3d90f
  -Accept:application/json Tus-Resumable:1.0.0 -a admin:secret
```

python-requests

```python
requests.head('http://nohost/plone/folder/@tus-upload/032803b64ad746b3ab46d9223ea3d90f',
  headers={'Accept': 'application/json', 'Tus-Resumable': '1.0.0'}, auth=('admin', 'secret'))
```

The server will respond with a 200: Ok status and the current file offset in the ‘Upload-Offset’ header:

```
HTTP/1.1 200 OK
Tus-Resumable: 1.0.0
Upload-Length: 8
Upload-Offset: 3
```

### 30.6 Configuration and Options

The current TUS configuration and a list of supported options can be retrieved sending an OPTIONS request to the @tus-upload endpoint:

```bash
curl

```

```bash
curl -i -X OPTIONS http://nohost/plone/folder/@tus-upload -H 'Accept: application/json' --user admin:secret
```

httpie

```bash
```

python-requests

```python
requests.options('http://nohost/plone/folder/@tus-upload', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

The server will respond with a 204: No content status and HTTP headers containing information about the available extentions and the TUS version:

```
HTTP/1.1 204 No Content
Tus-Extension: creation,expiration
Tus-Resumable: 1.0.0
Tus-Version: 1.0.0
```
30.7 CORS Configuration

If you use CORS and want to make it work with TUS, you have to make sure the TUS specific HTTP headers are allowed by your CORS policy.

```xml
<plone:CORSPolicy
  allow_origin="http://localhost"
  allow_methods="DELETE,GET,OPTIONS,PATCH,POST,PUT"
  allow_credentials="true"
  allow_headers="Accept,Authorization,Origin,X-Requested-With,Content-Type,Upload-Length,Upload-Offset,Tus-Resumable,Upload-Metadata"
  expose_headers="Upload-Offset,Location,Upload-Length,Tus-Version,Tus-Resumable,Tus-Max-Size,Tus-Extension,Upload-Metadata"
  max_age="3600"
/>
```

See the plone.rest documentation for more information on how to configure CORS policies.

See http://tus.io/protocols/resumable-upload.html#headers for a list and description of the individual headers.

30.8 Temporary Upload Directory

During upload files are stored in a temporary directory that by default is located in the CLIENT_HOME directory. If you are using a multi ZEO client setup without session stickiness you must configure this to a directory shared by all ZEO clients by setting the TUS_TMP_FILE_DIR environment variable. E.g. TUS_TMP_FILE_DIR=/tmp/tus-uploads
Vocabularies and Sources

Vocabularies are a set of allowed choices that back a particular field. They contain so called terms which represent those allowed choices. Sources are a similar, but are a more generic and dynamic concept.

31.1 Concepts

Vocabularies contain a list of terms. These terms are usually tokenized, meaning that in addition to a term’s value, it also has a token which is a machine-friendly identifier for the term (7bit ASCII).

**Note:** Since the underlying value of a term might not necessarily be serializable (it could be an arbitrary Python object), `plone.restapi` only exposes and accepts tokens, and will transparently convert between tokens and values during serialization / deserialization. For this reason, the following endpoints only support tokenized vocabularies / sources, and they do not expose the terms’ values.

Terms can also have a title, which is intended to be the user-facing label for the term. For vocabularies or sources whose terms are only tokenized, but not titled, `plone.restapi` will fall back to using the token as the term title.

Sources are similar to vocabularies, but they tend to be more dynamic in nature, and are often used for larger sets of terms. They are also not registered with a global name like vocabularies, but are instead addressed via the field they are assigned to.

Query Sources are sources that are capable of being queried / searched. The source will then return only the subset of terms that match the query.

The use of such a source is usually a strong indication that no attempt should be made to enumerate the full set of terms, but instead the source should only be queried, by presenting the user with an autocomplete widget for example.

Both vocabularies and sources can be context-sensitive, meaning that they take the context into account and their contents may therefore change depending on the context they’re invoked on.

This section can only provide a basic overview of vocabularies and related concepts. For a more in-depth explanation please refer to the Plone documentation.
31.2 Endpoints overview

In plone.restapi these three concepts are exposed through three separate endpoints (described in more detail below):

- `@vocabularies/(vocab_name)`
- `@sources/(field_name)`
- `@querysources/(field_name) ?query= (search_query)`

While the `@vocabularies` and `@sources` endpoints allow to enumerate terms (and optionally have terms filtered server-side), the `@querysources` endpoint only allows for searching the respective source.

31.3 List all vocabularies

**GET (context) /@vocabularies**

To retrieve a list of all the available vocabularies, send a GET request to the `@vocabularies` endpoint: http

```
GET /plone/@vocabularies HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl
```
curl -i http://nohost/plone/@vocabularies -H 'Accept: application/json' --user admin:secret
```

httpie
```
http http://nohost/plone/@vocabularies Accept:application/json -a admin:secret
```

python-requests
```
requests.get('http://nohost/plone/@vocabularies', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

The response will include a list with the URL (`@id`) and the names (`title`) of all the available vocabularies in Plone:

```
HTTP/1.1 200 OK
Content-Type: application/json

[  
  {  
      "title": "plone.app.vocabularies.AvailableContentLanguages" 
  },  
  {  
      "@id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabularies.SupportedContentLanguages",  
      "title": "plone.app.vocabularies.SupportedContentLanguages" 
  },  
  {  
      "title": "plone.app.vocabularies.Roles" 
  }
]
```
(continues on next page)
31.3. List all vocabularies

```
"title": "plone.app.vocabularies.Roles"
],
[
Permissions",
  "title": "plone.app.vocabularies.Permissions"
],
[
AllowedContentTypes",
  "title": "plone.app.vocabularies.AllowedContentTypes"
],
[
AllowableContentTypes",
  "title": "plone.app.vocabularies.AllowableContentTypes"
],
[
PortalTypes",
  "title": "plone.app.vocabularies.PortalTypes"
],
[
ReallyUserFriendlyTypes",
  "title": "plone.app.vocabularies.ReallyUserFriendlyTypes"
],
[
UserFriendlyTypes",
  "title": "plone.app.vocabularies.UserFriendlyTypes"
],
[
Skins",
  "title": "plone.app.vocabularies.Skins"
],
[
Workflows",
  "title": "plone.app.vocabularies.Workflows"
],
[
WorkflowStates",
  "title": "plone.app.vocabularies.WorkflowStates"
],
[
WorkflowTransitions",
  "title": "plone.app.vocabularies.WorkflowTransitions"
],
[
AvailableEditors",
  "title": "plone.app.vocabularies.AvailableEditors"
]
```

(continues on next page)
```json
{
    "@id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabularies.Keywords",
    "title": "plone.app.vocabularies.Keywords"
},
{
    "@id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabularies.SyndicationFeedTypes",
    "title": "plone.app.vocabularies.SyndicationFeedTypes"
},
{
    "@id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabularies.SyndicatableFeedItems",
    "title": "plone.app.vocabularies.SyndicatableFeedItems"
},
{
    "title": "plone.app.vocabularies.Users"
},
{
    "title": "plone.app.vocabularies.Groups"
},
{
    "title": "plone.app.vocabularies.Principals"
},
{
    "title": "plone.app.vocabularies.Catalog"
},
{
    "title": "plone.app.vocabularies.Actions"
},
{
    "title": "plone.app.vocabularies.PortalActionCategories"
},
{
    "title": "plone.app.vocabularies.Timezones"
},
{
    "title": "plone.app.vocabularies.CommonTimezones"
},
{
    "title": "plone.app.vocabularies.AvailableTimezones"
}
```
31.3. List all vocabularies

```json
{
    "title": "plone.app.vocabularies.Weekdays",
}
{
    "title": "plone.app.vocabularies.WeekdaysAbbr",
}
{
    "title": "plone.app.vocabularies.WeekdaysShort",
}
{
    "@id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabularies.Month",
    "title": "plone.app.vocabularies.Month",
}
{
    "title": "plone.app.vocabularies.MonthAbbr",
}
{
    "title": "plone.app.vocabularies.ImagesScales",
}
{
    "title": "plone.app.vocabularies.MetadataFields",
}
{
    "@id": "http://localhost:55001/plone/@vocabularies/plone.content.ValidAddableTypes",
    "title": "plone.app.content.ValidAddableTypes",
}
{
    "@id": "http://localhost:55001/plone/@vocabularies/plone.contentrules.events",
    "title": "plone.contentrules.events",
}
{
    "@id": "http://localhost:55001/plone/@vocabularies/Behaviors",
    "title": "Behaviors",
}
{
    "@id": "http://localhost:55001/plone/@vocabularies/Fields",
    "title": "Fields",
}
{
    "title": "plone.schemaeditor.VocabulariesVocabulary",
}
}```
"@id": "http://localhost:55001/plone/@vocabularies/plone.formwidget.relations.cmfcontentsearch",
"title": "plone.formwidget.relations.cmfcontentsearch"
},
"title": "plone.app.event.SynchronizationStrategies"
},
"@id": "http://localhost:55001/plone/@vocabularies/plone.app.contenttypes.metadatafields",
"title": "plone.app.contenttypes.metadatafields"
},
"@id": "http://localhost:55001/plone/@vocabularies/Interfaces",
"title": "Interfaces"
},
"title": "plone.app.discussion.vocabularies.CaptchaVocabulary"
},
"title": "plone.app.discussion.vocabularies.TextTransformVocabulary"
},
"@id": "http://localhost:55001/plone/@vocabularies/plone.app.users.user_registration_fields",
"title": "plone.app.users.user_registration_fields"
},
"@id": "http://localhost:55001/plone/@vocabularies/plone.app.users.group_ids",
"title": "plone.app.users.group_ids"
},
"@id": "http://localhost:55001/plone/@vocabularies/plone.app.multilingual.vocabularies.AllContentLanguageVocabulary",
"title": "plone.app.multilingual.vocabularies.AllContentLanguageVocabulary"
},
"@id": "http://localhost:55001/plone/@vocabularies/plone.app.multilingual.vocabularies.AllAvailableLanguageVocabulary",
"title": "plone.app.multilingual.vocabularies.AllAvailableLanguageVocabulary"
},
"title": "plone.app.multilingual.RootCatalog"
},
"@id": "http://localhost:55001/plone/@vocabularies/plone.restapi.testing.context_vocabulary",
"title": "plone.restapi.testing.context_vocabulary"
31.4 Get a vocabulary

GET  (context) /@vocabularies/
    vocab_name

To enumerate the terms of a particular vocabulary, use the @vocabularies endpoint with the name of the vocabulary, e.g. /plone/@vocabularies/plone.app.vocabularies.ReallyUserFriendlyTypes. The endpoint can be used with the site root and content objects. http

GET /plone/@vocabularies/plone.app.vocabularies.ReallyUserFriendlyTypes HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0

The server will respond with a list of terms. The title is purely for display purposes. The token is what should be sent to the server to address that term.

Note: Vocabulary terms will be batched if the size of the resultset exceeds the batch size. See Batching for more details on how to work with batched results.
31.4.1 Filter Vocabularies

GET (context) /@vocabularies/ vocab_name?title=filter_query

GET (context) /@vocabularies/ vocab_name?token=filter_query

Vocabulary terms can be filtered using the title or token parameter.

Use the title parameter to filter vocabulary terms by title. E.g. search for all terms that contain the string doc in the title: http
Use the `token` parameter to filter vocabulary terms by token. This is useful in case that you have the token and you need to retrieve the `title`. E.g. search the term `doc` in the token: http


HTTP/1.1 200 OK
Content-Type: application/json

```json
{
  "items": [
    {
      "title": "DX Test Document",
      "token": "DXTestDocument"
    },
    {
      "title": "Test Document",
      "token": "ATTestDocument"
    }
  ],
  "items_total": 2
}
```

31.4. Get a vocabulary
Note: You must not filter by title and token at the same time. The API returns a 400 response code if you do so.

### 31.5 Get a source

**HTTP** (`GET`)

**Method:** `GET`


**Accept:** `application/json`

**Response:**

```json
{
    "items": [
        {
            "title": "Page",
            "token": "Document"
        }
    ],
    "items_total": 1
}
```

**Note:** You must not filter by title and token at the same time. The API returns a 400 response code if you do so.

---

**31.5 Get a source**

**GET** `/@sources/` **field_name**

To enumerate the terms of a field’s source, use the `@sources` endpoint on a specific context, and pass the field name as a path parameter, e.g. `/plone/doc/@sources/some_field`.

Because sources are inherently tied to a specific field, this endpoint can only be invoked on content objects, and the source is addressed via the field name its used for, instead of a global name (which sources don’t have).

Otherwise the endpoint behaves the same as the `@vocabularies` endpoint.

Example: http

```text
GET /plone/doc/@sources/test_choice_with_source HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

**curl**

```bash
```

**httpie**

```bash
```
The server will respond with a list of terms. The title is purely for display purposes. The token is what should be sent to the server to address that term.

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "@id": "http://localhost:55001/plone/doc/@sources/test_choice_with_source",
  "items": [
    {
      "title": "Title 1",
      "token": "token1"
    },
    {
      "title": "Title 2",
      "token": "token2"
    },
    {
      "title": "Title 3",
      "token": "token3"
    }
  ],
  "items_total": 3
}
```

**Note:** Technically there can be sources that are not iterable (ones that only implement ISource, but not IIterableSource). These cannot be enumerated using the @sources endpoint, and it will respond with a corresponding error.

### 31.6 Querying a query source

**GET** *(context)*/@querysources/

`field_name?query=search_query`

Query sources (sources implementing IQuerySource) can be queried using this endpoint, by passing the search term in the `query` parameter. This search term will be passed to the query source’s `search()` method, and the source’s results are returned.

Example: http

```
GET /plone/doc/@querysources/test_choice_with_querysource?query=2 HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
```

httpie

### 31.6. Querying a query source 239

requests.get('http://nohost/plone/doc/@querysources/test_choice_with_querysource?query=2', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))

The server will respond with a list of terms. The title is purely for display purposes. The token is what should be sent to the server to address that term.

HTTP/1.1 200 OK
Content-Type: application/json

{
    "items": [
        {
            "title": "Title 2",
            "token": "token2"
        }
    ],
    "items_total": 1
}

Note: Even though technically sources that implement IQuerySource are required to implement __iter__ as well (when strictly following the interface inheritance hierarchy), they usually are used in Plone in situations where their full contents shouldn’t or can’t be enumerated (imagine a source of all users, backed by a large LDAP, for example).

For this reason, plone.restapi takes the stance that the IQuerySource interface is a strong indication that this source should **only** be queried, and therefore doesn’t support enumeration of terms via the @querysources endpoint.

*(If the source does actually implement IIterableSource in addition to IQuerySource, it can still be enumerated via the @sources endpoint)*
Control panels in Plone allow you to configure the global site setup of a Plone site. The `@controlpanels` endpoint in plone.restapi allows you to list all existing control panels in a Plone site and to retrieve or edit the settings of a specific control panel.

Most of the settings in the Plone control panels are based on `plone.registry` (since Plone 5.x). Therefore you can also use the `@registry` endpoint to retrieve or manipulate site settings. The `@controlpanels` endpoint just gives developers are more a convenience way of accessing the settings and makes it easier to render control panels on the front-end.

Note: This is currently only implemented for Plone 5.

### 32.1 Listing Control Panels

A list of all existing control panels in the portal can be retrieved by sending a GET request to the `@controlpanels` endpoint: http

```
GET /plone/@controlpanels HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i http://nohost/plone/@controlpanels -H 'Accept: application/json' --user admin:secret
```

httpie

```
http http://nohost/plone/@controlpanels Accept:application/json -a admin:secret
```

python-requests
requests.get('http://nohost/plone/@controlpanels', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))

Response:

HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "@id": "http://localhost:55001/plone/@controlpanels/date-and-time",
    "group": "General",
    "title": "Date and Time"
  },
  {
    "@id": "http://localhost:55001/plone/@controlpanels/language",
    "group": "General",
    "title": "Language"
  },
  {
    "@id": "http://localhost:55001/plone/@controlpanels/mail",
    "group": "General",
    "title": "Mail"
  },
  {
    "@id": "http://localhost:55001/plone/@controlpanels/navigation",
    "group": "General",
    "title": "Navigation"
  },
  {
    "@id": "http://localhost:55001/plone/@controlpanels/search",
    "group": "General",
    "title": "Search"
  },
  {
    "@id": "http://localhost:55001/plone/@controlpanels/site",
    "group": "General",
    "title": "Site"
  },
  {
    "@id": "http://localhost:55001/plone/@controlpanels/socialmedia",
    "group": "General",
    "title": "Social Media"
  },
  {
    "@id": "http://localhost:55001/plone/@controlpanels/dexterity-types",
    "group": "Content",
    "title": "Dexterity Content Types"
  },
  {
    "@id": "http://localhost:55001/plone/@controlpanels/editing",
    "group": "Content",
    "title": "Editing"
  },
  {
    "@id": "http://localhost:55001/plone/@controlpanels/imaging",
    "group": "Content",
    (continues on next page)
The following fields are returned:

- **@id**: hypermedia link to the control panel
- **title**: the title of the control panel
- **group**: the group where the control panel should show up (e.g. General, Content, Users, Security, Advanced, Add-on Configuration)

### 32.2 Retrieve a single Control Panel

To retrieve a single control panel, send a GET request to the URL of the control panel: http

```
GET /plone/@controlpanels/editing HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

```
curl -i http://nohost/plone/@controlpanels/editing -H 'Accept: application/json' --user admin:secret
```

```
http http://nohost/plone/@controlpanels/editing Accept:application/json -a admin:secret
```

```
requests.get('http://nohost/plone/@controlpanels/editing', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "@id": "http://localhost:55001/plone/@controlpanels/editing",
    "data": {
        "available_editors": [...
```
"TinyMCE",
"None"
],
"default_editor": {
"title": "TinyMCE",
"token": "TinyMCE"
},
"enable_link_integrity_checks": true,
"ext_editor": false,
"lock_on_ttw_edit": true,
"subjects_of_navigation_root": false
},
"group": "Content",
"schema": {
"fieldsets": [
{
"behavior": "plone",
"fields": [
"available_editors",
"default_editor",
"ext_editor",
"enable_link_integrity_checks",
"lock_on_ttw_edit",
"subjects_of_navigation_root"
],
"id": "default",
"title": "Default"
}
],
"properties": {
"available_editors": {
"additionalItems": true,
"default": ["TinyMCE", "None"],
"description": "Available editors in the portal.",
"factory": "List",
"items": {
"description": "",
"factory": "Text line (String)",
"title": "",
"type": "string"
},
"title": "Available editors",
"type": "array",
"uniqueItems": false
},
"default_editor": {
"default": "TinyMCE",
"description": "Select the default wysiwyg editor. Users will be able to choose their own or select to use the site default.",
"factory": "Choice",
"title": "Default editor",
"type": "string",
"vocabulary": {
"@id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabulary.AvailableEditors"
The following fields are returned:

- `@id`: hypermedia link to the control panel
- `title`: title of the control panel
- `group`: group name of the control panel
- `schema`: JSON Schema of the control panel
• data: current values of the control panel

### 32.3 Updating a Control Panel with PATCH

To update the settings on a control panel send a PATCH request to control panel resource: http

```
PATCH /plone/@controlpanels/editing HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
    "default_editor": "CKeditor",
    "ext_editor": true
}
```

curl

```
curl -i -X PATCH http://nohost/plone/@controlpanels/editing -H 'Accept: application/
˓→json' -H 'Content-Type: application/json' --data-raw '{"default_editor": "CKeditor",
˓→"ext_editor": true}' --user admin:secret
```

httpie

```
echo '{
    "default_editor": "CKeditor",
    "ext_editor": true
}' | http PATCH http://nohost/plone/@controlpanels/editing Accept:application/json
˓→Content-Type:application/json -a admin:secret
```

python-requests

```
requests.patch('http://nohost/plone/@controlpanels/editing', headers={'Accept':
˓→'application/json', 'Content-Type': 'application/json'}, json={'default_editor':
˓→'CKeditor', 'ext_editor': True}, auth=('admin', 'secret'))
```

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

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

### 32.4 Control Panels not based on plone.registry

Control panel which are not based on plone.registry have a custom @controlpanels/:panel endpoint implementation.

#### 32.4.1 Dexterity Types

@controlpanels/dexterity-types is a custom control panel endpoint, that will allow you to add, remove and configure available Types.

Reading or writing Dexterity Content Types require the plone.schemaeditor.ManageSchemata permission.
<table>
<thead>
<tr>
<th>Verb</th>
<th>URL</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td><code>/@controlpanels/dexterity-types</code></td>
<td>List configurable content-types</td>
</tr>
<tr>
<td>POST</td>
<td><code>/@controlpanels/dexterity-types</code></td>
<td>Creates a new content-type</td>
</tr>
<tr>
<td>GET</td>
<td><code>/@controlpanels/dexterity-types/</code></td>
<td>Get the current state of the content-type</td>
</tr>
<tr>
<td>PATCH</td>
<td><code>/@controlpanels/dexterity-types/</code></td>
<td>Update the content-type details</td>
</tr>
<tr>
<td>DELETE</td>
<td><code>/@controlpanels/dexterity-types/</code></td>
<td>Remove content-type</td>
</tr>
</tbody>
</table>

### Listing Dexterity Content Types

To list the available content-types send a GET request to `@controlpanels/dexterity-types http`:

```
GET /plone/@controlpanels/dexterity-types HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

```bash
curl -i http://nohost/plone/@controlpanels/dexterity-types -H 'Accept: application/json' --user admin:secret
```

```bash
```

```python
requests.get('http://nohost/plone/@controlpanels/dexterity-types', headers={'Accept': application/json}, auth=('admin', 'secret'))
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "@id": "http://localhost:55001/plone/@controlpanels/dexterity-types",
    "data": {},
    "group": "Content",
    "items": [
        {
            "@id": "http://localhost:55001/plone/@controlpanels/dexterity-types/Collection",
            "@type": "Collection",
            "count": 0,
            "description": "",
            "id": "Collection",
            "meta_type": "Dexterity FTI",
            "title": "Collection"
        },
        {
            "@type": "Document",
        }
    ]
}
```

(continues on next page)
"count": 0,
"description": "",
"id": "Document",
"meta_type": "Dexterity FTI",
"title": "Page"
},
{
"@id": "http://localhost:55001/plone/@controlpanels/dexterity-types/Folder",
"@type": "Folder",
"count": 0,
"description": "",
"id": "Folder",
"meta_type": "Dexterity FTI",
"title": "Folder"
},
{
"@id": "http://localhost:55001/plone/@controlpanels/dexterity-types/Link",
"@type": "Link",
"count": 0,
"description": "",
"id": "Link",
"meta_type": "Dexterity FTI",
"title": "Link"
},
{
"@id": "http://localhost:55001/plone/@controlpanels/dexterity-types/File",
"@type": "File",
"count": 0,
"description": "Lets you upload a file to the site.",
"id": "File",
"meta_type": "Dexterity FTI",
"title": "File"
},
{
"@id": "http://localhost:55001/plone/@controlpanels/dexterity-types/Image",
"@type": "Image",
"count": 0,
"description": "Images can be referenced in pages or displayed in an album.",
"id": "Image",
"meta_type": "Dexterity FTI",
"title": "Image"
},
{
"@id": "http://localhost:55001/plone/@controlpanels/dexterity-types/News Item",
"@type": "News Item",
"count": 0,
"description": "",
"id": "News Item",
"meta_type": "Dexterity FTI",
"title": "News Item"
},
{
"@id": "http://localhost:55001/plone/@controlpanels/dexterity-types/Event",
"@type": "Event",
"count": 0,
"description": "Events can be shown in calendars.",
"id": "Event",
"meta_type": "Dexterity FTI",
"title": "Event"}
The following fields are returned:

- @id: hypermedia link to the control panel
- title: title of the control panel
- group: group name of the control panel
- schema: JSON Schema of the control panel
- data: current values of the control panel
- items: list of configurable content-types.

Creating a new Dexterity Type with POST

To create a new content-type, send a POST request to the `/@controlpanels/dexterity-types` endpoint. HTTP

```plaintext
POST /plone/@controlpanels/dexterity-types HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
    "description": "A custom content-type",
    "title": "My Custom Content Type"
}
```

curl

```
```
httpie

echo '{
  "description": "A custom content-type",
  "title": "My Custom Content Type"
}' | http POST http://nohost/plone/@controlpanels/dexterity-types Accept:application/json Content-Type:application/json -a admin:secret

requests.post('http://nohost/plone/@controlpanels/dexterity-types', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'description': 'A custom content-type', 'title': 'My Custom Content Type'}, auth=('admin', 'secret'))

Response:

HTTP/1.1 201 Created
Content-Type: application/json
Location: http://localhost:55001/plone/@controlpanels/dexterity-types/my_custom_content_type

{
  "@id": "http://localhost:55001/plone/@controlpanels/dexterity-types/my_custom_content_type",
  "data": {
    "allowed_content_types": [],
    "description": "A custom content-type",
    "filter_content_types": true,
    "plone.allowdiscussion": false,
    "plone.basic": false,
    "plone.categorization": false,
    "plone.collection": false,
    "plone.constraintypes": false,
    "plone.dublincore": true,
    "plone.eventattendees": false,
    "plone.eventbasic": false,
    "plone.eventcontact": false,
    "plone.eventlocation": false,
    "plone.eventrecurrence": false,
    "plone.excludefromnavigation": false,
    "plone.leadimage": false,
    "plone.locking": false,
    "plone.namefromfilename": false,
    "plone.namefromtitle": true,
    "plone.navigationroot": false,
    "plone.nextpreviousenabled": false,
    "plone.nextpreviousstoggle": false,
    "plone.ownership": false,
    "plone.publication": false,
    "plone.relateditems": false,
    "plone.restapi.tests.dxtypes.ITestAnnotationsBehavior": false,
    "plone.restapi.tests.dxtypes.ITestBehavior": false,
    "plone.richtext": false,
    "plone.shortname": false,
    "plone.tableofcontents": false,
    "plone.thumb_icon": false,
    "plone.translatable": false,
    "plone.versioning": false,
  }
}


```
"title": "My Custom Content Type",
"volto.blocks": false
},
"description": "A custom content-type",
"group": "Content",
"items": [],
"schema": {
"fieldsets": [
{
"behavior": "plone",
"fields": [
"title",
"description",
"allowed_content_types",
"filter_content_types"
],
"id": "default",
"title": "Default"
},
{
"behavior": "plone",
"fields": [
"plone.allowdiscussion",
"plone.basic",
"volto.blocks",
"plone.categorization",
"plone.collection",
"plone.publication",
"plone.dublincore",
"plone.eventattendees",
"plone.eventbasic",
"plone.eventcontact",
"plone.eventlocation",
"plone.eventrecurrence",
"plone.excludefromnavigation",
"plone.constraintypes",
"plone.leadimage",
"plone.locking",
"plone.translatable",
"plone.namefromfilename",
"plone.namefromtitle",
"plone.navigationroot",
"plone.nextpreviousenabled",
"plone.nextpreviousstoggle",
"plone.ownership",
"plone.relateditems",
"plone.richtext",
"plone.shortname",
"plone.tableofcontents",
"plone.restapi.tests.dxtypes.ITestAnnotationsBehavior",
"plone.restapi.tests.dxtypes.ITestBehavior",
"plone.thumb_icon",
"plone.versioning"
],
"id": "behaviors",
"title": "Behaviors"
}
```
"properties": {
  "allowed_content_types": {
    "additionalItems": true,
    "description": "",
    "factory": "Multiple Choice",
    "items": {
      "description": "",
      "factory": "Choice",
      "title": "",
      "type": "string",
      "vocabulary": {
      }
    },
    "title": "Allowed Content Types",
    "type": "array",
    "uniqueItems": true
  },
  "description": {
    "description": "",
    "factory": "Text",
    "title": "Description",
    "type": "string",
    "widget": "textarea"
  },
  "filter_content_types": {
    "choices": [
      ["none", null],
      ["all", null],
      ["some", null]
    ],
    "default": "none",
    "description": "Items of this type can act as a folder containing other items. What content types should be allowed inside?",
    "enum": ["none", "all", "some"],
    "enumNames": [null, null, null],
    "factory": "Choice",
    "title": "Filter Contained Types",
  }
},
(continues on next page)
"type": "string",
"vocabulary": {
  "@id": "http://localhost:55001/plone/@sources/filter_content_types"
},
"plone.allowdiscussion": {
  "description": "Allow discussion on this item",
  "factory": "Yes/No",
  "title": "Allow discussion",
  "type": "boolean"
},
"plone.basic": {
  "description": "Adds title and description fields.",
  "factory": "Yes/No",
  "title": "Basic metadata",
  "type": "boolean"
},
"plone.categorization": {
  "description": "Adds keywords and language fields.",
  "factory": "Yes/No",
  "title": "Categorization",
  "type": "boolean"
},
"plone.collection": {
  "description": "Adds collection behavior",
  "factory": "Yes/No",
  "title": "Collection",
  "type": "boolean"
},
"plone.constrainttypes": {
  "description": "Restrict the content types that can be added to folderish content",
  "factory": "Yes/No",
  "title": "Folder Addable Constrains",
  "type": "boolean"
},
"plone.dublincore": {
  "description": "Adds standard metadata fields (equals Basic metadata + Categorization + Effective range + Ownership)",
  "factory": "Yes/No",
  "title": "Dublin Core metadata",
  "type": "boolean"
},
"plone.eventattendees": {
  "description": "Attendees extension for Events.",
  "factory": "Yes/No",
  "title": "Event Attendees",
  "type": "boolean"
},
"plone.eventbasic": {
  "description": "Basic Event schema.",
  "factory": "Yes/No",
  "title": "Event Basic",
  "type": "boolean"
},
"plone.eventcontact": {
  "description": "Contact extension for Events.",
}
"factory": "Yes/No",
"title": "Event Contact",
"type": "boolean"
},
"plone.eventlocation": {
  "description": "Location extension for Events.",
  "factory": "Yes/No",
  "title": "Event Location",
  "type": "boolean"
},
"plone.eventrecurrence": {
  "description": "Recurrence extension for Events.",
  "factory": "Yes/No",
  "title": "Event Recurrence",
  "type": "boolean"
},
"plone.excludefromnavigation": {
  "description": "Allow items to be excluded from navigation",
  "factory": "Yes/No",
  "title": "Exclude From navigation",
  "type": "boolean"
},
"plone.leadimage": {
  "description": "Adds image and image caption fields",
  "factory": "Yes/No",
  "title": "Lead Image",
  "type": "boolean"
},
"plone.locking": {
  "description": "Locking support for dexterity",
  "factory": "Yes/No",
  "title": "Locking",
  "type": "boolean"
},
"plone.namefromfilename": {
  "description": "Automatically generate short URL name for content based on its primary field file name",
  "factory": "Yes/No",
  "title": "Name from file name",
  "type": "boolean"
},
"plone.namefromtitle": {
  "description": "Automatically generate short URL name for content based on its initial title",
  "factory": "Yes/No",
  "title": "Name from title",
  "type": "boolean"
},
"plone.navigationroot": {
  "description": "Make all items of this type a navigation root",
  "factory": "Yes/No",
  "title": "Navigation root",
  "type": "boolean"
},
"plone.nextpreviousenabled": {
  "description": "Enable next previous navigation for all items of this type",
  "factory": "Yes/No",
  "title": "Next Previous"
"title": "Next previous navigation",
"type": "boolean"
},
"plone.nextpreviousstoggle": {
"description": "Allow items to have next previous navigation enabled",
"factory": "Yes/No",
"title": "Next previous navigation toggle",
"type": "boolean"
},
"plone.ownership": {
"description": "Adds creator, contributor, and rights fields.",
"factory": "Yes/No",
"title": "Ownership",
"type": "boolean"
},
"plone.publication": {
"description": "Adds effective date and expiration date fields.",
"factory": "Yes/No",
"title": "Date range",
"type": "boolean"
},
"plone.relateditems": {
"description": "Adds the ability to assign related items",
"factory": "Yes/No",
"title": "Related items",
"type": "boolean"
},
"plone.restapi.tests.dxtypes.ITestAnnotationsBehavior": {
"description": "Schema-only behavior using annotations",
"factory": "Yes/No",
"title": "Test Annotations Behavior",
"type": "boolean"
},
"plone.restapi.tests.dxtypes.ITestBehavior": {
"description": "Schema-only behavior using attributes",
"factory": "Yes/No",
"title": "Test Behavior",
"type": "boolean"
},
"plone.richtext": {
"description": "Adds richtext behavior",
"factory": "Yes/No",
"title": "RichText",
"type": "boolean"
},
"plone.shortname": {
"description": "Gives the ability to rename an item from its edit form.",
"factory": "Yes/No",
"title": "Short name",
"type": "boolean"
},
"plone.tableofcontents": {
"description": "Adds a table of contents",
"factory": "Yes/No",
"title": "Table of contents",
"type": "boolean"
},
(continues on next page)
"plone.thumb_icon": {
    "description": "Options to suppress thumbs and/or icons and to override thumb size in listings, tables etc.",
    "factory": "Yes/No",
    "title": "Thumbs and icon handling",
    "type": "boolean"
},
"plone.translatable": {
    "description": "Make this content type multilingual aware",
    "factory": "Yes/No",
    "title": "Multilingual Support",
    "type": "boolean"
},
"plone.versioning": {
    "description": "Versioning support with CMFEditions",
    "factory": "Yes/No",
    "title": "Versioning",
    "type": "boolean"
},
"title": {
    "description": "",
    "factory": "Text line (String)",
    "title": "Type Name",
    "type": "string"
},
"volto.blocks": {
    "description": "Enables Volto Blocks support",
    "factory": "Yes/No",
    "title": "Blocks",
    "type": "boolean"
},
"required": [
    "title",
    "filter_content_types"
],
"type": "object"}

**Reading a Dexterity Type with GET**

After a successful POST, access the content-type by sending a GET request to the /@controlpanels/dexterity-types/{type-id}:

```bash
GET /plone/@controlpanels/dexterity-types/my_custom_content_type HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```bash
```

httpie
http http://nohost/plone/@controlpanels/dexterity-types/my_custom_content_type
   --Accept:application/json -a admin:secret

python-requests

requests.get('http://nohost/plone/@controlpanels/dexterity-types/my_custom_content_type', headers={'Accept': 'application/json'}, auth=('admin', 'secret'))

Response:

HTTP/1.1 200 OK
Content-Type: application/json

{
    "@id": "http://localhost:55001/plone/@controlpanels/dexterity-types/my_custom_content_type",
    "data": {
        "allowed_content_types": [],
        "description": "A custom content-type",
        "filter_content_types": true,
        "plone.allowdiscussion": false,
        "plone.basic": false,
        "plone.categorization": false,
        "plone.collection": false,
        "plone.constrainttypes": false,
        "plone.dublincore": true,
        "plone.eventattendees": false,
        "plone.eventbasic": false,
        "plone.eventcontact": false,
        "plone.eventlocation": false,
        "plone.eventrecurrence": false,
        "plone.excludefromnavigation": false,
        "plone.leadimage": false,
        "plone.locking": false,
        "plone.namefromfilename": false,
        "plone.namefromtitle": true,
        "plone.navigationroot": false,
        "plone.nextpreviousenabled": false,
        "plone.nextpreviousstoggle": false,
        "plone.ownership": false,
        "plone.publication": false,
        "plone.relateditems": false,
        "plone.restapi.tests.dxtypes.ITestAnnotationsBehavior": false,
        "plone.restapi.tests.dxtypes.ITestBehavior": false,
        "plone.richtext": false,
        "plone.shortname": false,
        "plone.tableofcontents": false,
        "plone.thumb_icon": false,
        "plone.translatable": false,
        "plone.versioning": false,
        "title": "My Custom Content Type",
        "volto.blocks": false
    },
    "description": "A custom content-type",
    "group": "Content",
    "items": [],
    "schema": {

(continues on next page)
"fieldsets": [
  {
    "behavior": "plone",
    "fields": [
      "title",
      "description",
      "allowed_content_types",
      "filter_content_types"
    ],
    "id": "default",
    "title": "Default"
  },
  {
    "behavior": "plone",
    "fields": [
      "plone.allowdiscussion",
      "plone.basic",
      "volto.blocks",
      "plone.categorization",
      "plone.collection",
      "plone.publication",
      "plone.dublincore",
      "plone.eventattendees",
      "plone.eventbasic",
      "plone.eventcontact",
      "plone.eventlocation",
      "plone.eventrecurrence",
      "plone.excludefromnavigation",
      "plone.constrainetypes",
      "plone.leadimage",
      "plone.locking",
      "plone.translatable",
      "plone.namefromfilename",
      "plone.namefromtitle",
      "plone.navigationroot",
      "plone.nextpreviousenabled",
      "plone.nextprevioustoggle",
      "plone.ownership",
      "plone.relateditems",
      "plone.richtext",
      "plone.shortname",
      "plone.tableofcontents",
      "plone.restapi.tests.dxtypes.ITestAnnotationsBehavior",
      "plone.restapi.tests.dxtypes.ITestBehavior",
      "plone.thumb_icon",
      "plone.versioning"
    ],
    "id": "behaviors",
    "title": "Behaviors"
  }
],
"properties": {
  "allowed_content_types": {
    "additionalItems": true,
    "description": "",
    "factory": "Multiple Choice",
    "items":
  } }
"description": "", "factory": "Choice", "title": "", "type": "string", "vocabulary": {
  "$id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabularies.RealUserFriendlyTypes"
},
"title": "Allowed Content Types", "type": "array", "uniqueItems": true
},
"description": {
  "description": "", "factory": "Text", "title": "Description", "type": "string", "widget": "textarea"
},
"filter_content_types": {
  "choices": [
    ["none", null],
    ["all", null],
    ["some", null]
  ],
  "default": "none",
  "description": "Items of this type can act as a folder containing other items. What content types should be allowed inside?",
  "enum": [
    "none",
    "all",
    "some"
  ],
  "enumNames": [null, null, null],
  "factory": "Choice", "title": "Filter Contained Types", "type": "string", "vocabulary": {
    "$id": "http://localhost:55001/plone/@sources/filter_content_types"
  }
},
"plone.allowdiscussion": {
  "description": "Allow discussion on this item",
}}
"factory": "Yes/No",
"title": "Allow discussion",
"type": "boolean"
},
"plone.basic": {
"description": "Adds title and description fields.",
"factory": "Yes/No",
"title": "Basic metadata",
"type": "boolean"
},
"plone.categorization": {
"description": "Adds keywords and language fields.",
"factory": "Yes/No",
"title": "Categorization",
"type": "boolean"
},
"plone.collection": {
"description": "Adds collection behavior",
"factory": "Yes/No",
"title": "Collection",
"type": "boolean"
},
"plone.constraintypes": {
"description": "Restrict the content types that can be added to folderish content",
"factory": "Yes/No",
"title": "Folder Addable Constrains",
"type": "boolean"
},
"plone.dublincore": {
"description": "Adds standard metadata fields (equals Basic metadata + Categorization + Effective range + Ownership)",
"factory": "Yes/No",
"title": "Dublin Core metadata",
"type": "boolean"
},
"plone.eventattendees": {
"description": "Attendees extension for Events.",
"factory": "Yes/No",
"title": "Event Attendees",
"type": "boolean"
},
"plone.eventbasic": {
"description": "Basic Event schema.",
"factory": "Yes/No",
"title": "Event Basic",
"type": "boolean"
},
"plone.eventcontact": {
"description": "Contact extension for Events.",
"factory": "Yes/No",
"title": "Event Contact",
"type": "boolean"
},
"plone.eventlocation": {
"description": "Location extension for Events.",
"factory": "Yes/No",
"title": "Event location type.",
"type": "boolean"
"title": "Event Location",
"type": "boolean"
},
"plone.eventrecurrence": {
"description": "Recurrence extension for Events.",
"factory": "Yes/No",
"title": "Event Recurrence",
"type": "boolean"
},
"plone.excludefromnavigation": {
"description": "Allow items to be excluded from navigation",
"factory": "Yes/No",
"title": "Exclude From navigation",
"type": "boolean"
},
"plone.leadimage": {
"description": "Adds image and image caption fields",
"factory": "Yes/No",
"title": "Lead Image",
"type": "boolean"
},
"plone.locking": {
"description": "Locking support for dexterity",
"factory": "Yes/No",
"title": "Locking",
"type": "boolean"
},
"plone.namefromfilename": {
"description": "Automatically generate short URL name for content based on its primary field file name",
"factory": "Yes/No",
"title": "Name from file name",
"type": "boolean"
},
"plone.namefromtitle": {
"description": "Automatically generate short URL name for content based on its initial title",
"factory": "Yes/No",
"title": "Name from title",
"type": "boolean"
},
"plone.navigationroot": {
"description": "Make all items of this type a navigation root",
"factory": "Yes/No",
"title": "Navigation root",
"type": "boolean"
},
"plone.nextpreviousenabled": {
"description": "Enable next previous navigation for all items of this type",
"factory": "Yes/No",
"title": "Next previous navigation",
"type": "boolean"
},
"plone.nextprevioustoggle": {
"description": "Allow items to have next previous navigation enabled",
"factory": "Yes/No",
"title": "Next previous navigation toggle"
"type": "boolean"
},
"plone.ownership": {
  "description": "Adds creator, contributor, and rights fields.",
  "factory": "Yes/No",
  "title": "Ownership",
  "type": "boolean"
},
"plone.publication": {
  "description": "Adds effective date and expiration date fields.",
  "factory": "Yes/No",
  "title": "Date range",
  "type": "boolean"
},
"plone.relateditems": {
  "description": "Adds the ability to assign related items",
  "factory": "Yes/No",
  "title": "Related items",
  "type": "boolean"
},
"plone.restapi.tests.dxtypes.ITestAnnotationsBehavior": {
  "description": "Schema-only behavior using annotations",
  "factory": "Yes/No",
  "title": "Test Annotations Behavior",
  "type": "boolean"
},
"plone.restapi.tests.dxtypes.ITestBehavior": {
  "description": "Schema-only behavior using attributes",
  "factory": "Yes/No",
  "title": "Test Behavior",
  "type": "boolean"
},
"plone.richtext": {
  "description": "Adds richtext behavior",
  "factory": "Yes/No",
  "title": "RichText",
  "type": "boolean"
},
"plone.shortname": {
  "description": "Gives the ability to rename an item from its edit form.",
  "factory": "Yes/No",
  "title": "Short name",
  "type": "boolean"
},
"plone.tableofcontents": {
  "description": "Adds a table of contents",
  "factory": "Yes/No",
  "title": "Table of contents",
  "type": "boolean"
},
"plone.thumb_icon": {
  "description": "Options to suppress thumbs and/or icons and to override thumb size in listings, tables etc.",
  "factory": "Yes/No",
  "title": "Thumbs and icon handling",
  "type": "boolean"
}
Updating a Dexterity Type with PATCH

To update an existing content-type 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). http

```plaintext
PATCH /plone/@controlpanels/dexterity-types/my_custom_content_type HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
    "description": "A content-type",
    "plone.richtext": true,
    "plone.versioning": true,
    "title": "My Content Type"
}
```

curl


(continues on next page)
httpie

echo '{
  "description": "A content-type",
  "plone.richtext": true,
  "plone.versioning": true,
  "title": "My Content Type"
}' | http PATCH http://nohost/plone/@controlpanels/dexterity-types/my_custom_content_type
→ type Accept:application/json Content-Type:application/json -a admin:secret

python-requests

requests.patch('http://nohost/plone/@controlpanels/dexterity-types/my_custom_content_type', headers={'Accept': 'application/json', 'Content-Type': 'application/json'},
→ json={'description': 'A content-type', 'plone.richtext': True, 'plone.versioning': True,
→ 'title': 'My Content Type'}, auth=('admin', 'secret'))

Response:

HTTP/1.1 204 No Content

Removing a Dexterity Type with DELETE

Delete an existing content-type by sending a DELETE request to the URL of an existing content-type: http

DELETE /plone/@controlpanels/dexterity-types/my_custom_content_type HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0

curl

curl -i -X DELETE http://nohost/plone/@controlpanels/dexterity-types/my_custom_content_type
→ -H 'Accept: application/json' --user admin:secret

httpie

http DELETE http://nohost/plone/@controlpanels/dexterity-types/my_custom_content_type
→ -Accept:application/json -a admin:secret

python-requests

requests.delete('http://nohost/plone/@controlpanels/dexterity-types/my_custom_content_type',
→ headers=('Accept': 'application/json'), auth=('admin', 'secret'))

Response:

HTTP/1.1 204 No Content
CHAPTER 33

Tiles

Note: The tiles endpoint currently match only partially (the GET endpoints) the default Plone implementation.

A tile in Plone is an HTML snippet that can contain arbitrary content (e.g. text, images, videos).

33.1 Listing available tiles

Note: This endpoint currently does not return any data. The functionality needs to be implemented.

List all available tiles type by sending a GET request to the @tiles endpoint on the portal root:

GET /plone/@tiles HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0

The server responds with a Status 200 and list all available tiles:

HTTP/1.1 200 OK
Content-Type: application/json
[
    {
        "@id": "http://localhost:55001/plone/@tiles/title",
        "title": "Title tile",
        "description": "A field tile that will show the title of the content object",
    },
    {
        "@id": "http://localhost:55001/plone/@tiles/description",
        "title": "Description tile",
        "description": "A field tile that will show the description of the content object",
    }
]
33.2 Retrieve JSON schema of an individual tile

Note: This endpoint currently does not return any data. The functionality needs to be implemented.

Retrieve the JSON schema of a specific tile by calling the '@tiles' endpoint with the id of the tile:

```
GET /plone/@tiles/title HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

The server responds with a JSON schema definition for that particular tile:

```
HTTP/1.1 200 OK
Content-Type: application/json+schema

{
  "properties": {
    "title": {
      "description": "",
      "title": "Title",
      "type": "string"
    },
    ...,
    "required": [
      "title",
      "title": "Title Tile",
      "type": "object"
    ]
  }
}
```
The @querystring endpoint returns the querystring config of `plone.app.querystring`.

Instead of simply exposing the querystring related field and operation entries from the registry, it serializes them the same way the `@@querybuilderjsonconfig` view from `p.a.querystring` does.

This form is structured in a more convenient way for frontends to process:

- **Vocabularies** will be resolved, and their values will be inlined in the `values` property
- **Operations** will be inlined as well. The `operations` property will contain the list of operations (dotted names), and the `operators` property will contain the full definition of each of those operations supported by that field.
- Indexes that are flagged as sortable are listed in a dedicated top-level property `sortable_indexes`

Available options for the querystring in a Plone site can be queried by interacting with the `@querystring` endpoint on portal root:

### 34.1 Querystring Config

To retrieve all querystring options in the portal, call the `@querystring` endpoint with a GET request: http

```plaintext
GET /plone/@querystring HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```plaintext
curl -i http://nohost/plone/@querystring -H 'Accept: application/json' --user admin:secret
```

httpie
http http://nohost/plone/@querystring Accept:application/json -a admin:secret

python-requests

def requests.get('http://nohost/plone/@querystring', headers={'Accept': 'application/json'}, auth=('admin', 'secret')):

The server will respond with all querystring options in the portal:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "@id": "http://localhost:55001/plone/@querystring",
  "indexes": {
    "Creator": {
      "description": "The person that created an item",
      "enabled": true,
      "group": "Metadata",
      "operations": [
        "plone.app.querystring.operation.string.currentUser",
        "plone.app.querystring.operation.selection.any"
      ],
      "operators": {
        "plone.app.querystring.operation.selection.any": {
          "description": "Tip: you can use * to autocomplete."
        },
        "plone.app.querystring.operation.string.currentUser": {
          "description": "The user viewing the querystring results",
          "title": "Current logged in user",
          "widget": "null"
        }
      },
      "sortable": true,
      "title": "Creator",
      "values": {},
      "vocabulary": "plone.app.vocabularies.Users"
    },
    "Description": {
      "description": "An item's description",
      "enabled": true,
      "group": "Text",
      "operations": ["plone.app.querystring.operation.string.contains"
      ],
      "operators": {
        "plone.app.querystring.operation.string.contains": {
          "description": null,
          "title": "Contains",
          "widget": "StringWidget"
        }
      }
    }
  }
}
```

(continues on next page)
"sortable": false,
"title": "Description",
"values": {},
"vocabulary": null
},
"SearchableText": {
"description": "Text search of an item's contents",
"enabled": true,
"group": "Text",
"operations": [
"plone.app.querystring.operation.string.contains"
],
"operators": {
"plone.app.querystring.operation.string.contains": {
"description": null,
"operation": "plone.app.querystring.queryparser._contains",
"title": "Contains",
"widget": "StringWidget"
}
},
"sortable": false,
"title": "Searchable text",
"values": {},
"vocabulary": null
},
"Subject": {
"description": "Tags are used for organization of content",
"enabled": true,
"group": "Text",
"operations": [
"plone.app.querystring.operation.selection.any",
"plone.app.querystring.operation.selection.all"
],
"operators": {
"plone.app.querystring.operation.selection.all": {
"description": "Tip: you can use * to autocomplete.",
"operation": "plone.app.querystring.queryparser._all",
"title": "Matches all of",
"widget": "MultipleSelectionWidget"
},
"plone.app.querystring.operation.selection.any": {
"description": "Tip: you can use * to autocomplete.",
"operation": "plone.app.querystring.queryparser._contains",
"title": "Matches any of",
"widget": "MultipleSelectionWidget"
}
},
"sortable": false,
"title": "Tag",
"values": {},
"vocabulary": "plone.app.vocabularies.Keywords"
},
"Title": {
"description": "Text search of an item's title",
"enabled": true,
"group": "Text",
"operations": [

(continues on next page)
"plone.app.querystring.operation.string.contains",

"operators": {
  "plone.app.querystring.operation.string.contains": {
    "description": null,
    "operation": "plone.app.querystring.queryparser._contains",
    "title": "Contains",
    "widget": "StringWidget"
  }
},
"sortable": false,
"title": "Title",
"values": {},
"vocabulary": null},

"created": {
  "description": "The date an item was created",
  "enabled": true,
  "group": "Dates",
  "operations": [
    "plone.app.querystring.operation.date.lessThan",
    "plone.app.querystring.operation.date.largerThan",
    "plone.app.querystring.operation.date.between",
    "plone.app.querystring.operation.date.lessThanRelativeDate",
    "plone.app.querystring.operation.date.largerThanRelativeDate",
    "plone.app.querystring.operation.date.today",
    "plone.app.querystring.operation.date.beforeToday",
    "plone.app.querystring.operation.date.afterToday",
    "plone.app.querystring.operation.date.beforeRelativeDate",
    "plone.app.querystring.operation.date.afterRelativeDate"
  ],
  "operators": {
    "plone.app.querystring.operation.date.afterRelativeDate": {
      "description": "After N days in the future",
      "operation": "plone.app.querystring.queryparser._afterRelativeDate",
      "title": "After relative Date",
      "widget": "RelativeDateWidget"
    },
    "plone.app.querystring.operation.date.afterToday": {
      "description": "After the current day",
      "operation": "plone.app.querystring.queryparser._afterToday",
      "title": "After today",
      "widget": null
    },
    "plone.app.querystring.operation.date.beforeRelativeDate": {
      "description": "Before N days in the past",
      "operation": "plone.app.querystring.queryparser._beforeRelativeDate",
      "title": "Before relative Date",
      "widget": "RelativeDateWidget"
    },
    "plone.app.querystring.operation.date.beforeToday": {
      "description": "Before the current day",
      "operation": "plone.app.querystring.queryparser._beforeToday",
      "title": "Before today",
      "widget": null
    },
    "plone.app.querystring.operation.date.between": {
      "description": "Between two dates",
      "operation": "plone.app.querystring.queryparser._between",
      "title": "Between dates",
      "widget": "RelativeDateWidget"
    }
  }
}
"description": "Please use YYYY/MM/DD.",
"operation": "plone.app.querystring.queryparser._between",
"title": "Between dates",
"widget": "DateRangeWidget"
},
"plone.app.querystring.operation.date.largerThan": {
"description": "Please use YYYY/MM/DD.",
"operation": "plone.app.querystring.queryparser._largerThan",
"title": "After date",
"widget": "DateWidget"
},
"plone.app.querystring.operation.date.largerThanRelativeDate": {
"description": "Please enter the number in days.",
"operation": "plone.app.querystring.queryparser._moreThanRelativeDate",
"title": "Within last",
"widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.lessThan": {
"description": "Please use YYYY/MM/DD.",
"operation": "plone.app.querystring.queryparser._lessThan",
"title": "Before date",
"widget": "DateWidget"
},
"plone.app.querystring.operation.date.lessThanRelativeDate": {
"description": "Please enter the number in days.",
"operation": "plone.app.querystring.queryparser._lessThanRelativeDate",
"title": "Within next",
"widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.today": {
"description": "The current day",
"operation": "plone.app.querystring.queryparser._today",
"title": "Today",
"widget": null
}
},
"sortable": true,
"title": "Creation date",
"values": {},
"vocabulary": null
},
"effective": {
"description": "The time and date an item was first published",
"enabled": true,
"group": "Dates",
"operations": [
"plone.app.querystring.operation.date.lessThan",
"plone.app.querystring.operation.date.largerThan",
"plone.app.querystring.operation.date.between",
"plone.app.querystring.operation.date.lessThanRelativeDate",
"plone.app.querystring.operation.date.largerThanRelativeDate",
"plone.app.querystring.operation.date.today",
"plone.app.querystring.operation.date.beforeToday",
"plone.app.querystring.operation.date.afterToday",
"plone.app.querystring.operation.date.beforeRelativeDate",
"plone.app.querystring.operation.date.afterRelativeDate"
]}
}
"operators": {
    "plone.app.querystring.operation.date.afterRelativeDate": {
        "description": "After N days in the future",
        "operation": "plone.app.querystring.queryparser._afterRelativeDate",
        "title": "After relative Date",
        "widget": "RelativeDateWidget"
    },
    "plone.app.querystring.operation.date.afterToday": {
        "description": "After the current day",
        "operation": "plone.app.querystring.queryparser._afterToday",
        "title": "After today",
        "widget": null
    },
    "plone.app.querystring.operation.date.beforeRelativeDate": {
        "description": "Before N days in the past",
        "operation": "plone.app.querystring.queryparser._beforeRelativeDate",
        "title": "Before relative Date",
        "widget": "RelativeDateWidget"
    },
    "plone.app.querystring.operation.date.beforeToday": {
        "description": "Before the current day",
        "operation": "plone.app.querystring.queryparser._beforeToday",
        "title": "Before today",
        "widget": null
    },
    "plone.app.querystring.operation.date.between": {
        "description": "Please use YYYY/MM/DD.",
        "operation": "plone.app.querystring.queryparser._between",
        "title": "Between dates",
        "widget": "DateRangeWidget"
    },
    "plone.app.querystring.operation.date.largerThan": {
        "description": "Please use YYYY/MM/DD.",
        "operation": "plone.app.querystring.queryparser._largerThan",
        "title": "After date",
        "widget": "DateWidget"
    },
    "plone.app.querystring.operation.date.largerThanRelativeDate": {
        "description": "Please enter the number in days.",
        "operation": "plone.app.querystring.queryparser._moreThanRelativeDate",
        "title": "Within last",
        "widget": "RelativeDateWidget"
    },
    "plone.app.querystring.operation.date.lessThan": {
        "description": "Please use YYYY/MM/DD.",
        "operation": "plone.app.querystring.queryparser._lessThan",
        "title": "Before date",
        "widget": "DateWidget"
    },
    "plone.app.querystring.operation.date.lessThanRelativeDate": {
        "description": "Please enter the number in days.",
        "operation": "plone.app.querystring.queryparser._lessThanRelativeDate",
        "title": "Within next",
        "widget": "RelativeDateWidget"
    },
    "plone.app.querystring.operation.date.today": {
        "description": "The current day",
        "operation": "plone.app.querystring.queryparser._today",
        "title": "Today",
        "widget": null
    }
}
"operation": "plone.app.querystring.queryparser._today",
"title": "Today",
"widget": null
}
],
"sortable": true,
"title": "Effective date",
"values": {},
"vocabulary": null
},
"effectiveRange": {
"description": "Querying this is undefined",
"enabled": false,
"group": "Dates",
"operations": [],
"operators": {},
"sortable": false,
"title": "Effective range",
"values": {},
"vocabulary": null
},
"end": {
"description": "The end date and time of an event",
"enabled": true,
"group": "Dates",
"operations": [
"plone.app.querystring.operation.date.lessThan",
"plone.app.querystring.operation.date.largerThan",
"plone.app.querystring.operation.date.between",
"plone.app.querystring.operation.date.lessThanRelativeDate",
"plone.app.querystring.operation.date.largerThanRelativeDate",
"plone.app.querystring.operation.date.today",
"plone.app.querystring.operation.date.beforeToday",
"plone.app.querystring.operation.date.afterToday",
"plone.app.querystring.operation.date.beforeRelativeDate",
"plone.app.querystring.operation.date.afterRelativeDate"
],
"operators": {
"plone.app.querystring.operation.date.afterRelativeDate": {
"description": "After N days in the future",
"operation": "plone.app.querystring.queryparser._afterRelativeDate",
"title": "After relative Date",
"widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.afterToday": {
"description": "After the current day",
"operation": "plone.app.querystring.queryparser._afterToday",
"title": "After today",
"widget": null
},
"plone.app.querystring.operation.date.beforeRelativeDate": {
"description": "Before N days in the past",
"operation": "plone.app.querystring.queryparser._beforeRelativeDate",
"title": "Before relative Date",
"widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.beforeToday": {
..."}
"description": "Before the current day",
"operation": "plone.app.querystring.queryparser._beforeToday",
"title": "Before today",
"widget": null
},
"plone.app.querystring.operation.date.between": {
  "description": "Please use YYYY/MM/DD."
  "operation": "plone.app.querystring.queryparser._between",
  "title": "Between dates",
  "widget": "DateRangeWidget"
},
"plone.app.querystring.operation.date.largerThan": {
  "description": "Please use YYYY/MM/DD."
  "operation": "plone.app.querystring.queryparser._largerThan",
  "title": "After date",
  "widget": "DateWidget"
},
"plone.app.querystring.operation.date.largerThanRelativeDate": {
  "description": "Please enter the number in days."
  "operation": "plone.app.querystring.queryparser._moreThanRelativeDate",
  "title": "Within last",
  "widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.lessThan": {
  "description": "Please use YYYY/MM/DD."
  "operation": "plone.app.querystring.queryparser._lessThan",
  "title": "Before date",
  "widget": "DateWidget"
},
"plone.app.querystring.operation.date.lessThanRelativeDate": {
  "description": "Please enter the number in days."
  "operation": "plone.app.querystring.queryparser._lessThanRelativeDate",
  "title": "Within next",
  "widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.today": {
  "description": "The current day"
  "operation": "plone.app.querystring.queryparser._today",
  "title": "Today",
  "widget": null
},
"sortable": true,
"title": "Event end date",
"values": {},
"vocabulary": null
},
"expires": {
  "description": "The time and date an item was expired",
  "enabled": true,
  "group": "Dates",
  "operations": [
    "plone.app.querystring.operation.date.lessThan",
    "plone.app.querystring.operation.date.largerThan",
    "plone.app.querystring.operation.date.between",
    "plone.app.querystring.operation.date.lessThanRelativeDate",
    "plone.app.querystring.operation.date.largerThanRelativeDate"
  ]
}
"plone.app.querystring.operation.date.today",
"plone.app.querystring.operation.date.beforeToday",
"plone.app.querystring.operation.date.afterToday",
"plone.app.querystring.operation.date.beforeRelativeDate",
"plone.app.querystring.operation.date.afterRelativeDate"
],

"operators": {
  "plone.app.querystring.operation.date.afterRelativeDate": {
    "description": "After N days in the future",
    "operation": "plone.app.querystring.queryparser._afterRelativeDate",
    "title": "After relative Date",
    "widget": "RelativeDateWidget"
  },
  "plone.app.querystring.operation.date.afterToday": {
    "description": "After the current day",
    "operation": "plone.app.querystring.queryparser._afterToday",
    "title": "After today",
    "widget": null
  },
  "plone.app.querystring.operation.date.beforeRelativeDate": {
    "description": "Before N days in the past",
    "operation": "plone.app.querystring.queryparser._beforeRelativeDate",
    "title": "Before relative Date",
    "widget": "RelativeDateWidget"
  },
  "plone.app.querystring.operation.date.beforeToday": {
    "description": "Before the current day",
    "operation": "plone.app.querystring.queryparser._beforeToday",
    "title": "Before today",
    "widget": null
  },
  "plone.app.querystring.operation.date.between": {
    "description": "Please use YYYY/MM/DD.",
    "operation": "plone.app.querystring.queryparser._between",
    "title": "Between dates",
    "widget": "DateRangeWidget"
  },
  "plone.app.querystring.operation.date.largerThan": {
    "description": "Please use YYYY/MM/DD.",
    "operation": "plone.app.querystring.queryparser._largerThan",
    "title": "After date",
    "widget": "DateWidget"
  },
  "plone.app.querystring.operation.date.largerThanRelativeDate": {
    "description": "Please enter the number in days.",
    "operation": "plone.app.querystring.queryparser._moreThanRelativeDate",
    "title": "Within last",
    "widget": "RelativeDateWidget"
  },
  "plone.app.querystring.operation.date.lessThan": {
    "description": "Please use YYYY/MM/DD.",
    "operation": "plone.app.querystring.queryparser._lessThan",
    "title": "Before date",
    "widget": "DateWidget"
  },
  "plone.app.querystring.operation.date.lessThanRelativeDate": {
    "description": "Please enter the number in days.",
    "operation": "plone.app.querystring.queryparser._lessThanRelativeDate",
    "title": "Within last",
    "widget": "RelativeDateWidget"
  }
}
"operation": "plone.app.querystring.queryparser._lessThanRelativeDate",
"title": "Within next",
"widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.today": {
  "description": "The current day",
  "operation": "plone.app.querystring.queryparser._today",
  "title": "Today",
  "widget": null
},
"sortable": true,
"title": "Expiration date",
"values": {},
"vocabulary": null
},
"getId": {
  "description": "The short name of an item (used in the url)",
  "enabled": true,
  "group": "Metadata",
  "operations": [
    "plone.app.querystring.operation.string.is"
  ],
  "operators": {
    "plone.app.querystring.operation.string.is": {
      "description": "Tip: you can use * to autocomplete.",
      "operation": "plone.app.querystring.queryparser._equal",
      "title": "Is",
      "widget": "StringWidget"
    }
  },
  "sortable": true,
  "title": "Short name (id)",
  "values": {},
  "vocabulary": null
},
"getObjPositionInParent": {
  "description": "The order of an item in its parent folder",
  "enabled": false,
  "group": "Metadata",
  "operations": [
    "plone.app.querystring.operation.int.is",
    "plone.app.querystring.operation.int.lessThan",
    "plone.app.querystring.operation.int.largerThan"
  ],
  "operators": {
    "plone.app.querystring.operation.int.is": {
      "description": null,
      "operation": "plone.app.querystring.queryparser._intEqual",
      "title": "Equals",
      "widget": "StringWidget"
    },
    "plone.app.querystring.operation.int.largerThan": {
      "description": null,
      "operation": "plone.app.querystring.queryparser._intLargerThan",
      "title": "Larger than",
      "widget": "StringWidget"
    }
  }
}
"plone.app.querystring.operation.int.lessThan": {
  "description": null,
  "operation": "plone.app.querystring.queryparser._intLessThan",
  "title": "Less than",
  "widget": "StringWidget"
},

"sortable": true,
"title": "Order in folder",
"values": {},
"vocabulary": null
},

"getRawRelatedItems": {
  "description": "Find items related to the selected items",
  "enabled": false,
  "group": "Metadata",
  "operators": [
    "plone.app.querystring.operation.reference.is"
  ],
  "sortable": false,
  "title": "Related To",
  "values": {},
  "vocabulary": null
},

"isDefaultPage": {
  "description": "Find items that are the default view for their containing
  folder.",
  "enabled": false,
  "group": "Metadata",
  "operators": [
    "plone.app.querystring.operation.boolean.isTrue",
    "plone.app.querystring.operation.boolean.isFalse"
  ],
  "sortable": false,
  "title": "Default Page"
}

"title": "Default Page",
"values": {},
"vocabulary": null
},

"isFolderish": {
"description": "Find items that can contain other objects",
"enabled": false,
"group": "Metadata",
"operations": [
"plone.app.querystring.operation.boolean.isTrue",
"plone.app.querystring.operation.boolean.isFalse"
],

"operators": {
"plone.app.querystring.operation.boolean.isFalse": {
"description": null,
"operation": "plone.app.querystring.queriesparser._isFalse",
"title": "No",
"widget": null
},

"plone.app.querystring.operation.boolean.isTrue": {
"description": null,
"operation": "plone.app.querystring.queriesparser._isTrue",
"title": "Yes",
"widget": null
}
},

"sortable": false,
"title": "Folder-like",
"values": {},
"vocabulary": null
},

"modified": {
"description": "The time and date an item was last modified",
"enabled": true,
"group": "Dates",
"operations": [
"plone.app.querystring.operation.date.lessThan",
"plone.app.querystring.operation.date.largerThan",
"plone.app.querystring.operation.date.between",
"plone.app.querystring.operation.date.lessThanRelativeDate",
"plone.app.querystring.operation.date.largerThanRelativeDate",
"plone.app.querystring.operation.date.today",
"plone.app.querystring.operation.date.beforeToday",
"plone.app.querystring.operation.date.afterToday",
"plone.app.querystring.operation.date.beforeRelativeDate",
"plone.app.querystring.operation.date.afterRelativeDate"
],

"operators": {
"plone.app.querystring.operation.date.afterRelativeDate": {
"description": "After N days in the future",
"operation": "plone.app.querystring.queriesparser._afterRelativeDate",
"title": "After Date",
"widget": "RelativeDateWidget"
},

"plone.app.querystring.operation.date.afterToday": {
"description": "After the current day",
"operation": "plone.app.querystring.queriesparser._afterToday",
"title": "After the current day",
"widget": "AfterTodayWidget"
}
"title": "After today",
"widget": null
},
"plone.app.querystring.operation.date.beforeRelativeDate": {
"description": "Before N days in the past",
"operation": "plone.app.querystring.queryparser._beforeRelativeDate",
"title": "Before relative Date",
"widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.beforeToday": {
"description": "Before the current day",
"operation": "plone.app.querystring.queryparser._beforeToday",
"title": "Before today",
"widget": null
},
"plone.app.querystring.operation.date.between": {
"description": "Please use YYYY/MM/DD.",
"operation": "plone.app.querystring.queryparser._between",
"title": "Between dates",
"widget": "DateRangeWidget"
},
"plone.app.querystring.operation.date.largerThan": {
"description": "Please use YYYY/MM/DD.",
"operation": "plone.app.querystring.queryparser._largerThan",
"title": "After date",
"widget": "DateWidget"
},
"plone.app.querystring.operation.date.largerThanRelativeDate": {
"description": "Please enter the number in days.",
"operation": "plone.app.querystring.queryparser._moreThanRelativeDate",
"title": "Within last",
"widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.lessThan": {
"description": "Please use YYYY/MM/DD.",
"operation": "plone.app.querystring.queryparser._lessThan",
"title": "Before date",
"widget": "DateWidget"
},
"plone.app.querystring.operation.date.lessThanRelativeDate": {
"description": "Please enter the number in days.",
"operation": "plone.app.querystring.queryparser._lessThanRelativeDate",
"title": "Within next",
"widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.today": {
"description": "The current day",
"operation": "plone.app.querystring.queryparser._today",
"title": "Today",
"widget": null
},
"sortable": true,
"title": "Modification date",
"values": {},
"vocabulary": null}
"path": {
  "description": "The location of an item ",
  "enabled": true,
  "group": "Metadata",
  "operations": [
    "plone.app.querystring.operation.string.absolutePath",
    "plone.app.querystring.operation.string.path",
    "plone.app.querystring.operation.string.relativePath"
  ],
  "operators": {
    "plone.app.querystring.operation.string.absolutePath": {
      "description": "Location in the site structure",
      "operation": "plone.app.querystring.queryparser._absolutePath",
      "title": "Absolute path",
      "widget": "ReferenceWidget"
    },
    "plone.app.querystring.operation.string.path": {
      "description": "Location in the navigation structure",
      "operation": "plone.app.querystring.queryparser._navigationPath",
      "title": "Navigation path",
      "widget": "ReferenceWidget"
    },
    "plone.app.querystring.operation.string.relativePath": {
      "description": "Use '../' to navigate to parent objects.",
      "operation": "plone.app.querystring.queryparser._relativePath",
      "title": "Relative path",
      "widget": "RelativePathWidget"
    }
  },
  "sortable": false,
  "title": "Location",
  "values": {},
  "vocabulary": null
},
"portal_type": {
  "description": "An item's type (e.g. Event)",
  "enabled": true,
  "group": "Metadata",
  "operations": [
    "plone.app.querystring.operation.selection.any"
  ],
  "operators": {
    "plone.app.querystring.operation.selection.any": {
      "description": "Tip: you can use * to autocomplete.",
      "operation": "plone.app.querystring.queryparser._contains",
      "title": "Matches any of",
      "widget": "MultipleSelectionWidget"
    }
  },
  "sortable": false,
  "title": "Type",
  "values": {
    "ATTestDocument": {
      "title": "Test Document"
    },
    "ATTestFolder": {
      "title": "Test Folder"
    }
  }
}
"Collection": {
  "title": "Collection"
},
"DXTestDocument": {
  "title": "DX Test Document"
},
"Discussion Item": {
  "title": "Comment"
},
"Document": {
  "title": "Page"
},
"Event": {
  "title": "Event"
},
"File": {
  "title": "File"
},
"Folder": {
  "title": "Folder"
},
"Image": {
  "title": "Image"
},
"Link": {
  "title": "Link"
},
"News Item": {
  "title": "News Item"
},
"vocabulary": "plone.app.vocabularies.ReallyUserFriendlyTypes"
},
"review_state": {
  "description": "An item's workflow state (e.g. published)",
  "enabled": true,
  "group": "Metadata",
  "operations": [
    "plone.app.querystring.operation.selection.any"
  ],
  "operators": {
    "plone.app.querystring.operation.selection.any": {
      "description": "Tip: you can use * to autocomplete.",
      "operation": "plone.app.querystring.queryparser._contains",
      "title": "Matches any of",
      "widget": "MultipleSelectionWidget"
    }
  },
  "sortable": true,
  "title": "Review state",
  "values": {
    "external": {
      "title": "Externally visible [external]"
    },
    "internal": {
      "title": "Internal draft [internal]"
    }
  }
}
"internally_published": {  
  "title": "Internally published [internally_published]"
},
"pending": {  
  "title": "Pending [pending]"
},
"private": {  
  "title": "Private [private]"
},
"published": {  
  "title": "Published with accent \u00e9 [published]"
},
"rejected": {  
  "title": "Rejected [rejected]"
},
"spam": {  
  "title": "Spam [spam]"
},
"visible": {  
  "title": "Public draft [visible]"
}
},
"vocabulary": "plone.app.vocabularies.WorkflowStates"
},
"show_inactive": {  
  "description": "Select which roles have the permission to view inactive objects"
},
"enabled": true,
"group": "Metadata",
"operations": [  
  "plone.app.querystring.operation.string.showInactive"
],
"operators": {  
  "plone.app.querystring.operation.string.showInactive": {  
    "description": "The user roles which are allowed to see inactive content",
    "operation": "plone.app.querystring.queryparser._showInactive",
    "title": "Show Inactive",
    "widget": "MultipleSelectionWidget"
  }
},
"sortable": false,
"title": "Show Inactive",
"values": {  
  "Anonymous": {  
    "title": "Anonymous"
  },
  "Authenticated": {  
    "title": "Authenticated"
  },
  "Contributor": {  
    "title": "Contributor"
  },
  "Editor": {  
    "title": "Editor"
  },
  "Manager": {
(continues on next page)
"title": "Manager"
},
"Member": {
    "title": "Member"
},
"Owner": {
    "title": "Owner"
},
"Reader": {
    "title": "Reader"
},
"Reviewer": {
    "title": "Reviewer"
},
"Site Administrator": {
    "title": "Site Administrator"
},
"vocabulary": "plone.app.vocabularies.Roles"
},
"sortable_title": {
    "description": "The item's title, transformed for sorting",
    "enabled": false,
    "group": "Text",
    "operations": [
        "plone.app.querystring.operation.string.contains",
        "plone.app.querystring.operation.string.is"
    ],
    "operators": {
        "plone.app.querystring.operation.string.contains": {
            "description": null,
            "operation": "plone.app.querystring.queryparser._contains",
            "title": "Contains",
            "widget": "StringWidget"
        },
        "plone.app.querystring.operation.string.is": {
            "description": "Tip: you can use * to autocomplete.",
            "operation": "plone.app.querystring.queryparser._equal",
            "title": "Is",
            "widget": "StringWidget"
        }
    },
    "sortable": true,
    "title": "Sortable Title",
    "values": {},
    "vocabulary": null
},
"start": {
    "description": "The start date and time of an event",
    "enabled": true,
    "group": "Dates",
    "operations": [
        "plone.app.querystring.operation.date.lessThan",
        "plone.app.querystring.operation.date.largerThan",
        "plone.app.querystring.operation.date.between",
        "plone.app.querystring.operation.date.lessThanRelativeDate",
        "plone.app.querystring.operation.date.largerThanRelativeDate"
    ]}
"plone.app.querystring.operation.date.today",
"plone.app.querystring.operation.date.beforeToday",
"plone.app.querystring.operation.date.afterToday",
"plone.app.querystring.operation.date.beforeRelativeDate",
"plone.app.querystring.operation.date.afterRelativeDate"
],
"operators": {
"plone.app.querystring.operation.date.afterRelativeDate": {
"description": "After N days in the future",
"operation": "plone.app.querystring.queryparser._afterRelativeDate",
"title": "After relative Date",
"widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.afterToday": {
"description": "After the current day",
"operation": "plone.app.querystring.queryparser._afterToday",
"title": "After today",
"widget": null
},
"plone.app.querystring.operation.date.beforeRelativeDate": {
"description": "Before N days in the past",
"operation": "plone.app.querystring.queryparser._beforeRelativeDate",
"title": "Before relative Date",
"widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.beforeToday": {
"description": "Before the current day",
"operation": "plone.app.querystring.queryparser._beforeToday",
"title": "Before today",
"widget": null
},
"plone.app.querystring.operation.date.between": {
"description": "Please use YYYY/MM/DD.",
"operation": "plone.app.querystring.queryparser._between",
"title": "Between dates",
"widget": "DateRangeWidget"
},
"plone.app.querystring.operation.date.largerThan": {
"description": "Please use YYYY/MM/DD.",
"operation": "plone.app.querystring.queryparser._largerThan",
"title": "After date",
"widget": "DateWidget"
},
"plone.app.querystring.operation.date.largerThanRelativeDate": {
"description": "Please enter the number in days.",
"operation": "plone.app.querystring.queryparser._moreThanRelativeDate",
"title": "Within last",
"widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.lessThan": {
"description": "Please use YYYY/MM/DD.",
"operation": "plone.app.querystring.queryparser._lessThan",
"title": "Before date",
"widget": "DateWidget"
},
"plone.app.querystring.operation.date.lessThanRelativeDate": {
"description": "Please enter the number in days.",
"operator": "plone.app.querystring.queryparser._lessThanRelativeDate",
"title": "Within last",
"widget": "RelativeDateWidget"
}
"operation": "plone.app.querystring.queryparser._lessThanRelativeDate",
"title": "Within next",
"widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.today": {
"description": "The current day",
"operation": "plone.app.querystring.queryparser._today",
"title": "Today",
"widget": null
}
},
"sortable": true,
"title": "Event start date",
"values": {},
"vocabulary": null
},
"sortable_indexes": {
"Creator": {
"description": "The person that created an item",
"enabled": true,
"group": "Metadata",
"operations": [
"plone.app.querystring.operation.string.currentUser",
"plone.app.querystring.operation.selection.any"
],
"operators": {
"plone.app.querystring.operation.selection.any": {
"description": "Tip: you can use * to autocomplete.",
"operation": "plone.app.querystring.queryparser._contains",
"title": "Matches any of",
"widget": "MultipleSelectionWidget"
},
"plone.app.querystring.operation.string.currentUserId": {
"description": "The user viewing the querystring results",
"operation": "plone.app.querystring.queryparser._currentUserId",
"title": "Current logged in user",
"widget": null
}
},
"sortable": true,
"title": "Creator",
"values": {},
"vocabulary": "plone.app.vocabularies.Users"
},
"created": {
"description": "The date an item was created",
"enabled": true,
"group": "Dates",
"operations": [
"plone.app.querystring.operation.date.lessThan",
"plone.app.querystring.operation.date.largerThan",
"plone.app.querystring.operation.date.between",
"plone.app.querystring.operation.date.lessThanRelativeDate",
"plone.app.querystring.operation.date.largerThanRelativeDate",
"plone.app.querystring.operation.date.today",
"plone.app.querystring.operation.date.beforeToday",
"plone.app.querystring.operation.date.afterToday"
],
"operators": {
"plone.app.querystring.operation.selection.any": {
"description": "Tip: you can use * to autocomplete.",
"operation": "plone.app.querystring.queryparser._contains",
"title": "Matches any of",
"widget": "MultipleSelectionWidget"
},
"plone.app.querystring.operation.string.currentUserId": {
"description": "The user viewing the querystring results",
"operation": "plone.app.querystring.queryparser._currentUserId",
"title": "Current logged in user",
"widget": null
}
},
"sortable": true,
"title": "Created",
"values": {},
"vocabulary": null
}
"plone.app.querystring.operation.date.afterToday",
"plone.app.querystring.operation.date.beforeRelativeDate",
"plone.app.querystring.operation.date.afterRelativeDate",

],
"operators": {
"plone.app.querystring.operation.date.afterRelativeDate": {
    "description": "After N days in the future",
    "operation": "plone.app.querystring.queryparser._afterRelativeDate",
    "title": "After relative Date",
    "widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.afterToday": {
    "description": "After the current day",
    "operation": "plone.app.querystring.queryparser._afterToday",
    "title": "After today",
    "widget": null
},
"plone.app.querystring.operation.date.beforeRelativeDate": {
    "description": "Before N days in the past",
    "operation": "plone.app.querystring.queryparser._beforeRelativeDate",
    "title": "Before relative Date",
    "widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.beforeToday": {
    "description": "Before the current day",
    "operation": "plone.app.querystring.queryparser._beforeToday",
    "title": "Before today",
    "widget": null
},
"plone.app.querystring.operation.date.between": {
    "description": "Please use YYYY/MM/DD.",
    "operation": "plone.app.querystring.queryparser._between",
    "title": "Between dates",
    "widget": "DateRangeWidget"
},
"plone.app.querystring.operation.date.largerThan": {
    "description": "Please use YYYY/MM/DD.",
    "operation": "plone.app.querystring.queryparser._largerThan",
    "title": "After date",
    "widget": "DateWidget"
},
"plone.app.querystring.operation.date.largerThanRelativeDate": {
    "description": "Please enter the number in days.",
    "operation": "plone.app.querystring.queryparser._moreThanRelativeDate",
    "title": "Within last",
    "widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.lessThan": {
    "description": "Please use YYYY/MM/DD.",
    "operation": "plone.app.querystring.queryparser._lessThan",
    "title": "Before date",
    "widget": "DateWidget"
},
"plone.app.querystring.operation.date.lessThanRelativeDate": {
    "description": "Please enter the number in days.",
    "operation": "plone.app.querystring.queryparser._lessThanRelativeDate",
    "title": "Within next",
    "widget": "RelativeDateWidget"
}
"widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.today": {
  "description": "The current day",
  "operation": "plone.app.querystring.queryparser._today",
  "title": "Today",
  "widget": null
}
",
"sortable": true,
"title": "Creation date",
"values": {},
"vocabulary": null
}
,"effective": {
  "description": "The time and date an item was first published",
  "enabled": true,
  "group": "Dates",
  "operations": [
    "plone.app.querystring.operation.date.lessThan",
    "plone.app.querystring.operation.date.largerThan",
    "plone.app.querystring.operation.date.between",
    "plone.app.querystring.operation.date.lessThanRelativeDate",
    "plone.app.querystring.operation.date.largerThanRelativeDate",
    "plone.app.querystring.operation.date.today",
    "plone.app.querystring.operation.date.beforeToday",
    "plone.app.querystring.operation.date.afterToday",
    "plone.app.querystring.operation.date.beforeRelativeDate",
    "plone.app.querystring.operation.date.afterRelativeDate"
  ],
  "operators": {
    "plone.app.querystring.operation.date.afterRelativeDate": {
      "description": "After N days in the future",
      "operation": "plone.app.querystring.queryparser._afterRelativeDate",
      "title": "After relative Date",
      "widget": "RelativeDateWidget"
    },
    "plone.app.querystring.operation.date.afterToday": {
      "description": "After the current day",
      "operation": "plone.app.querystring.queryparser._afterToday",
      "title": "After today",
      "widget": null
    },
    "plone.app.querystring.operation.date.beforeRelativeDate": {
      "description": "Before N days in the past",
      "operation": "plone.app.querystring.queryparser._beforeRelativeDate",
      "title": "Before relative Date",
      "widget": "RelativeDateWidget"
    },
    "plone.app.querystring.operation.date.beforeToday": {
      "description": "Before the current day",
      "operation": "plone.app.querystring.queryparser._beforeToday",
      "title": "Before today",
      "widget": null
    },
    "plone.app.querystring.operation.date.between": {
      "description": "Please use YYYY/MM/DD."
    }
  }
}
"operation": "plone.app.querystring.queryparser._between",
"title": "Between dates",
"widget": "DateRangeWidget"
},
"plone.app.querystring.operation.date.largerThan": {
"description": "Please use YYYY/MM/DD.",
"operation": "plone.app.querystring.queryparser._largerThan",
"title": "After date",
"widget": "DateWidget"
},
"plone.app.querystring.operation.date.largerThanRelativeDate": {
"description": "Please enter the number in days.",
"operation": "plone.app.querystring.queryparser._moreThanRelativeDate",
"title": "Within last",
"widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.lessThan": {
"description": "Please use YYYY/MM/DD.",
"operation": "plone.app.querystring.queryparser._lessThan",
"title": "Before date",
"widget": "DateWidget"
},
"plone.app.querystring.operation.date.lessThanRelativeDate": {
"description": "Please enter the number in days.",
"operation": "plone.app.querystring.queryparser._lessThanRelativeDate",
"title": "Within next",
"widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.today": {
"description": "The current day",
"operation": "plone.app.querystring.queryparser._today",
"title": "Today",
"widget": null
}
],
"sortable": true,
"title": "Effective date",
"values": {},
"vocabulary": null
},
"end": {
"description": "The end date and time of an event",
"enabled": true,
"group": "Dates",
"operations": [
"plone.app.querystring.operation.date.lessThan",
"plone.app.querystring.operation.date.largerThan",
"plone.app.querystring.operation.date.between",
"plone.app.querystring.operation.date.lessThanRelativeDate",
"plone.app.querystring.operation.date.largerThanRelativeDate",
"plone.app.querystring.operation.date.today",
"plone.app.querystring.operation.date.beforeToday",
"plone.app.querystring.operation.date.afterToday",
"plone.app.querystring.operation.date.beforeRelativeDate",
"plone.app.querystring.operation.date.afterRelativeDate"
]
},
"operators": {
"plone.app.querystring.operation.date.afterRelativeDate": {  
  "description": "After N days in the future",  
  "operation": "plone.app.querystring.queryparser._afterRelativeDate",  
  "title": "After relative Date",  
  "widget": "RelativeDateWidget" 
},
"plone.app.querystring.operation.date.afterToday": {  
  "description": "After the current day",  
  "operation": "plone.app.querystring.queryparser._afterToday",  
  "title": "After today",  
  "widget": null 
},
"plone.app.querystring.operation.date.beforeRelativeDate": {  
  "description": "Before N days in the past",  
  "operation": "plone.app.querystring.queryparser._beforeRelativeDate",  
  "title": "Before relative Date",  
  "widget": "RelativeDateWidget" 
},
"plone.app.querystring.operation.date.beforeToday": {  
  "description": "Before the current day",  
  "operation": "plone.app.querystring.queryparser._beforeToday",  
  "title": "Before today",  
  "widget": null 
},
"plone.app.querystring.operation.date.between": {  
  "description": "Please use YYYY/MM/DD.",  
  "operation": "plone.app.querystring.queryparser._between",  
  "title": "Between dates",  
  "widget": "DateRangeWidget" 
},
"plone.app.querystring.operation.date.largerThan": {  
  "description": "Please use YYYY/MM/DD.",  
  "operation": "plone.app.querystring.queryparser._largerThan",  
  "title": "After date",  
  "widget": "DateWidget" 
},
"plone.app.querystring.operation.date.largerThanRelativeDate": {  
  "description": "Please enter the number in days.",  
  "operation": "plone.app.querystring.queryparser._moreThanRelativeDate",  
  "title": "Within last",  
  "widget": "RelativeDateWidget" 
},
"plone.app.querystring.operation.date.lessThan": {  
  "description": "Please use YYYY/MM/DD.",  
  "operation": "plone.app.querystring.queryparser._lessThan",  
  "title": "Before date",  
  "widget": "DateWidget" 
},
"plone.app.querystring.operation.date.lessThanRelativeDate": {  
  "description": "Please enter the number in days.",  
  "operation": "plone.app.querystring.queryparser._lessThanRelativeDate",  
  "title": "Within next",  
  "widget": "RelativeDateWidget" 
},
"plone.app.querystring.operation.date.today": {  
  "description": "The current day",  
  "operation": "plone.app.querystring.queryparser._today"}
"title": "Today",
  "widget": null
}
],
"sortable": true,
"title": "Event end date",
"values": {},
"vocabulary": null
},
"expires": {
  "description": "The time and date an item was expired",
  "enabled": true,
  "group": "Dates",
  "operations": [
    "plone.app.querystring.operation.date.lessThan",
    "plone.app.querystring.operation.date.largerThan",
    "plone.app.querystring.operation.date.between",
    "plone.app.querystring.operation.date.lessThanRelativeDate",
    "plone.app.querystring.operation.date.largerThanRelativeDate",
    "plone.app.querystring.operation.date.today",
    "plone.app.querystring.operation.date.beforeToday",
    "plone.app.querystring.operation.date.afterToday",
    "plone.app.querystring.operation.date.beforeRelativeDate",
    "plone.app.querystring.operation.date.afterRelativeDate"
  ],
  "operators": {
    "plone.app.querystring.operation.date.afterRelativeDate": {
      "description": "After N days in the future",
      "operation": "plone.app.querystring.queryparser._afterRelativeDate",
      "title": "After relative Date",
      "widget": "RelativeDateWidget"
    },
    "plone.app.querystring.operation.date.afterToday": {
      "description": "After the current day",
      "operation": "plone.app.querystring.queryparser._afterToday",
      "title": "After today",
      "widget": null
    },
    "plone.app.querystring.operation.date.beforeRelativeDate": {
      "description": "Before N days in the past",
      "operation": "plone.app.querystring.queryparser._beforeRelativeDate",
      "title": "Before relative Date",
      "widget": "RelativeDateWidget"
    },
    "plone.app.querystring.operation.date.beforeToday": {
      "description": "Before the current day",
      "operation": "plone.app.querystring.queryparser._beforeToday",
      "title": "Before today",
      "widget": null
    },
    "plone.app.querystring.operation.date.between": {
      "description": "Please use YYYY/MM/DD.",
      "operation": "plone.app.querystring.queryparser._between",
      "title": "Between dates",
      "widget": "DateRangeWidget"
    },
    "plone.app.querystring.operation.date.largerThan": {
      "description": "More than N days in the future",
      "operation": "plone.app.querystring.queryparser._largerThan",
      "title": "More than relative Date",
      "widget": "RelativeDateWidget"
    }
  ]
}
"description": "Please use YYYY/MM/DD.",
"operation": "plone.app.querystring.queryparser._largerThan",
"title": "After date",
"widget": "DateWidget"
},

"plone.app.querystring.operation.date.largerThanRelativeDate": {
  "description": "Please enter the number in days.",
  "operation": "plone.app.querystring.queryparser._moreThanRelativeDate",
  "title": "Within last",
  "widget": "RelativeDateWidget"
},

"plone.app.querystring.operation.date.lessThan": {
  "description": "Please use YYYY/MM/DD.",
  "operation": "plone.app.querystring.queryparser._lessThan",
  "title": "Before date",
  "widget": "DateWidget"
},

"plone.app.querystring.operation.date.lessThanRelativeDate": {
  "description": "Please enter the number in days.",
  "operation": "plone.app.querystring.queryparser._lessThanRelativeDate",
  "title": "Within next",
  "widget": "RelativeDateWidget"
},

"plone.app.querystring.operation.date.today": {
  "description": "The current day",
  "operation": "plone.app.querystring.queryparser._today",
  "title": "Today",
  "widget": null
}
},

"sortable": true,
"title": "Expiration date",
"values": {},
"vocabulary": null
},

"getId": {
  "description": "The short name of an item (used in the url)",
  "enabled": true,
  "group": "Metadata",
  "operations": [
    "plone.app.querystring.operation.string.is"
  ],
  "operators": {
    "plone.app.querystring.operation.string.is": {
      "description": "Tip: you can use * to autocomplete.",
      "operation": "plone.app.querystring.queryparser._equal",
      "title": "Is",
      "widget": "StringWidget"
    }
  },
  "sortable": true,
  "title": "Short name (id)",
  "values": {},
  "vocabulary": null
},

"getObjPositionInParent": {
  "description": "The order of an item in its parent folder",
  "enabled": true,
  "group": "Metadata",
  "operations": [
    "plone.app.querystring.operation.string.is"
  ],
  "operators": {
    "plone.app.querystring.operation.string.is": {
      "description": "Tip: you can use * to autocomplete.",
      "operation": "plone.app.querystring.queryparser._equal",
      "title": "Is",
      "widget": "StringWidget"
    }
  },
  "sortable": true,
  "title": "Short name (id)",
  "values": {},
  "vocabulary": null
},

"getObjPositionInParent": {
  "description": "The order of an item in its parent folder",
  "enabled": true,
  "group": "Metadata",
  "operations": [
    "plone.app.querystring.operation.string.is"
  ],
  "operators": {
    "plone.app.querystring.operation.string.is": {
      "description": "Tip: you can use * to autocomplete.",
      "operation": "plone.app.querystring.queryparser._equal",
      "title": "Is",
      "widget": "StringWidget"
    }
  },
  "sortable": true,
  "title": "Short name (id)",
  "values": {},
  "vocabulary": null
},

"getObjPositionInParent": {
"enabled": false,
"group": "Metadata",
"operations": [
  "plone.app.querystring.operation.int.is",
  "plone.app.querystring.operation.int.lessThan",
  "plone.app.querystring.operation.int.largerThan"
],
"operators": {
  "plone.app.querystring.operation.int.is": {
    "description": null,
    "operation": "plone.app.querystring.queryparser._intEqual",
    "title": "Equals",
    "widget": "StringWidget"
  },
  "plone.app.querystring.operation.int.largerThan": {
    "description": null,
    "operation": "plone.app.querystring.queryparser._intLargerThan",
    "title": "Larger than",
    "widget": "StringWidget"
  },
  "plone.app.querystring.operation.int.lessThan": {
    "description": null,
    "operation": "plone.app.querystring.queryparser._intLessThan",
    "title": "Less than",
    "widget": "StringWidget"
  }
},
"sortable": true,
"title": "Order in folder",
"values": {},
"vocabulary": null
},
"modified": {
  "description": "The time and date an item was last modified",
  "enabled": true,
  "group": "Dates",
  "operations": [
    "plone.app.querystring.operation.date.lessThan",
    "plone.app.querystring.operation.date.largerThan",
    "plone.app.querystring.operation.date.between",
    "plone.app.querystring.operation.date.lessThanRelativeDate",
    "plone.app.querystring.operation.date.largerThanRelativeDate",
    "plone.app.querystring.operation.date.today",
    "plone.app.querystring.operation.date.beforeToday",
    "plone.app.querystring.operation.date.afterToday",
    "plone.app.querystring.operation.date.beforeRelativeDate",
    "plone.app.querystring.operation.date.afterRelativeDate"
  ],
  "operators": {
    "plone.app.querystring.operation.date.afterRelativeDate": {
      "description": "After N days in the future",
      "operation": "plone.app.querystring.queryparser._afterRelativeDate",
      "title": "After relative Date",
      "widget": "RelativeDateWidget"
    },
    "plone.app.querystring.operation.date.afterToday": {
      "description": "After the current day",
      "operation": "plone.app.querystring.queryparser._afterToday",
      "title": "After the current Date",
      "widget": "DateWidget"
    }
  }
}


```
"operation": "plone.app.querystring.queryparser._afterToday",
"title": "After today",
"widget": null
},
"plone.app.querystring.operation.date.beforeRelativeDate": {
  "description": "Before N days in the past",
  "operation": "plone.app.querystring.queryparser._beforeRelativeDate",
  "title": "Before relative Date",
  "widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.beforeToday": {
  "description": "Before the current day",
  "operation": "plone.app.querystring.queryparser._beforeToday",
  "title": "Before today",
  "widget": null
},
"plone.app.querystring.operation.date.between": {
  "description": "Please use YYYY/MM/DD."
  "operation": "plone.app.querystring.queryparser._between",
  "title": "Between dates",
  "widget": "DateRangeWidget"
},
"plone.app.querystring.operation.date.largerThan": {
  "description": "Please use YYYY/MM/DD."
  "operation": "plone.app.querystring.queryparser._largerThan",
  "title": "After date",
  "widget": "DateWidget"
},
"plone.app.querystring.operation.date.largerThanRelativeDate": {
  "description": "Please enter the number in days."
  "operation": "plone.app.querystring.queryparser._moreThanRelativeDate",
  "title": "Within last",
  "widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.lessThan": {
  "description": "Please use YYYY/MM/DD."
  "operation": "plone.app.querystring.queryparser._lessThan",
  "title": "Before date",
  "widget": "DateWidget"
},
"plone.app.querystring.operation.date.lessThanRelativeDate": {
  "description": "Please enter the number in days."
  "operation": "plone.app.querystring.queryparser._lessThanRelativeDate",
  "title": "Within next",
  "widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.today": {
  "description": "The current day",
  "operation": "plone.app.querystring.queryparser._today",
  "title": "Today",
  "widget": null
}
```

(continues on next page)
"review_state": {  
    "description": "An item's workflow state (e.g. published)",  
    "enabled": true,  
    "group": "Metadata",  
    "operations": [  
        "plone.app.querystring.operation.selection.any"  
    ],  
    "operators": {  
        "plone.app.querystring.operation.selection.any": {  
            "description": "Tip: you can use * to autocomplete.",  
            "operation": "plone.app.querystring.queryparser._contains",  
            "title": "Matches any of",  
            "widget": "MultipleSelectionWidget"  
        }  
    },  
    "sortable": true,  
    "title": "Review state",  
    "values": {  
        "external": {  
            "title": "Externally visible [external]"  
        },  
        "internal": {  
            "title": "Internal draft [internal]"  
        },  
        "internally_published": {  
            "title": "Internally published [internally_published]"  
        },  
        "pending": {  
            "title": "Pending [pending]"  
        },  
        "private": {  
            "title": "Private [private]"  
        },  
        "published": {  
            "title": "Published with accent \u00e9 [published]"  
        },  
        "rejected": {  
            "title": "Rejected [rejected]"  
        },  
        "spam": {  
            "title": "Spam [spam]"  
        },  
        "visible": {  
            "title": "Public draft [visible]"  
        }  
    },  
    "vocabulary": "plone.app.vocabularies.WorkflowStates"  
},  
"sortable_title": {  
    "description": "The item's title, transformed for sorting",  
    "enabled": false,  
    "group": "Text",  
    "operations": [  
        "plone.app.querystring.operation.string.contains",  
        "plone.app.querystring.operation.string.is"  
    ]}
"operators": { 
  "plone.app.querystring.operation.string.contains": { 
    "description": null, 
    "operation": "plone.app.querystring.queryparser._contains", 
    "title": "Contains", 
    "widget": "StringWidget"
  },
  "plone.app.querystring.operation.string.is": { 
    "description": "Tip: you can use * to autocomplete.", 
    "operation": "plone.app.querystring.queryparser._equal", 
    "title": "Is", 
    "widget": "StringWidget"
  }
},
"sortable": true,
"title": "Sortable Title",
"values": {},
"vocabulary": null
},
"start": { 
  "description": "The start date and time of an event",
  "enabled": true,
  "group": "Dates",
  "operations": [ 
    "plone.app.querystring.operation.date.lessThan",
    "plone.app.querystring.operation.date.largerThan",
    "plone.app.querystring.operation.date.between",
    "plone.app.querystring.operation.date.lessThanRelativeDate",
    "plone.app.querystring.operation.date.largerThanRelativeDate",
    "plone.app.querystring.operation.date.today",
    "plone.app.querystring.operation.date.beforeToday",
    "plone.app.querystring.operation.date.afterToday",
    "plone.app.querystring.operation.date.beforeRelativeDate",
    "plone.app.querystring.operation.date.afterRelativeDate"
  ],
  "operators": { 
    "plone.app.querystring.operation.date.afterRelativeDate": { 
      "description": "After N days in the future",
      "operation": "plone.app.querystring.queryparser._afterRelativeDate",
      "title": "After relative Date",
      "widget": "RelativeDateWidget"
    },
    "plone.app.querystring.operation.date.afterToday": { 
      "description": "After the current day",
      "operation": "plone.app.querystring.queryparser._afterToday",
      "title": "After today",
      "widget": null
    },
    "plone.app.querystring.operation.date.beforeRelativeDate": { 
      "description": "Before N days in the past",
      "operation": "plone.app.querystring.queryparser._beforeRelativeDate",
      "title": "Before relative Date",
      "widget": "RelativeDateWidget"
    },
    "plone.app.querystring.operation.date.beforeToday": { 
      "description": "Before the current day",
      "operation": "plone.app.querystring.queryparser._beforeToday",
      "title": null
    }
  }
}
"title": "Before today",
"widget": null
},
"plone.app.querystring.operation.date.between": {
"description": "Please use YYYY/MM/DD.",
"operation": "plone.app.querystring.queryparser._between",
"title": "Between dates",
"widget": "DateRangeWidget"
},
"plone.app.querystring.operation.date.largerThan": {
"description": "Please use YYYY/MM/DD.",
"operation": "plone.app.querystring.queryparser._largerThan",
"title": "After date",
"widget": "DateWidget"
},
"plone.app.querystring.operation.date.largerThanRelativeDate": {
"description": "Please enter the number in days.",
"operation": "plone.app.querystring.queryparser._moreThanRelativeDate",
"title": "Within last",
"widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.lessThan": {
"description": "Please use YYYY/MM/DD.",
"operation": "plone.app.querystring.queryparser._lessThan",
"title": "Before date",
"widget": "DateWidget"
},
"plone.app.querystring.operation.date.lessThanRelativeDate": {
"description": "Please enter the number in days.",
"operation": "plone.app.querystring.queryparser._lessThanRelativeDate",
"title": "Within next",
"widget": "RelativeDateWidget"
},
"plone.app.querystring.operation.date.today": {
"description": "The current day",
"operation": "plone.app.querystring.queryparser._today",
"title": "Today",
"widget": null
},
"sortable": true,
"title": "Event start date",
"values": {},
"vocabulary": null
}
Querystring Search

The `@querystring-search` endpoint given a p.a.querystring query returns the results.

You can call the `/@querystring-search` endpoint with a POST request and the p.a.querystring query in JSON BODY, along with the others querystring options:

```
POST /plone/@querystring-search HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{  
    "query": [
        {
            "i": "portal_type",
            "o": "plone.app.querystring.operation.selection.any",
            "v": [
                "Document"
            ]
        }
    ]
}
```

curl

```
```

httpie

```
echo '{
  "query": [
  ]
}'
```

(continues on next page)
requests.post('http://nohost/plone/@querystring-search', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'query': [{'i': 'portal_type', 'o': 'plone.app.querystring.operation.selection.any', 'v': ['Document']}]}, auth=('admin', 'secret'))

The server will respond with the results:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "@id": "http://localhost:55001/plone/@querystring-search",
   "items": [
      {
         "@id": "http://localhost:55001/plone/front-page",
         "@type": "Document",
         "description": "Congratulations! You have successfully installed Plone.",
         "review_state": "private",
         "title": "Welcome to Plone"
      },
      {
         "@id": "http://localhost:55001/plone/testdocument",
         "@type": "Document",
         "description": "",
         "review_state": "private",
         "title": "Test Document"
      }
   ],
   "items_total": 2
}
```

The endpoint accepts the same parameters than the querystring search accept:

- `b_start`
- `b_size`
- `sort_on`
- `sort_order`: "ascending", "descending"
- `limit`

and also the `fullobjects` parameter for return the whole objects instead of the summary serialization of them.
CHAPTER 36

Customizing the API

36.1 Content serialization

36.1.1 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:

```python
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()
```

(continues on next page)
return json_compatible(value)

Register the adapter in ZCML:

```xml
<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`).
37.1 Naming Convention for REST API Resources/Endpoints

37.1.1 Nouns vs Verbs

Rule: Use nouns to represent resources.

Do:

/my-folder
/@registry
/@types

Don’t:

/createFolder
/deleteDocument
/updateEvent

Reason:

RESTful URI should refer to a resource that is a thing (noun) instead of referring to an action (verb) because nouns have properties as verbs do not. The REST architectural principle uses HTTP verbs to interact with resources.

Though, there is an exception to that rule, verbs can be used for specific actions or calculations, e.g.:

/login
/logout
/move-to
/reset-password

37.1.2 Singular vs Plural

Rule: Use plural resources.
Do:

/users
/users/21

Don’t:

/user
/user/21

Reason:
If you use singular for a collection like resource (e.g. “/user” to retrieve a list of all users) it feels wrong. Mixing singular and plural is confusing (e.g. user “/users” for retrieving users and “/user/21” to retrieve a single user).

37.1.3 Upper vs. Lowercase

Rule: Use lowercase letters in URIs.
Do:

http://example.com/my-folder/my-document

Don’t:

http://example.com/My-Folder/My-Document

Reason: RFC 3986 defines URIs as case-sensitive except for the scheme and host components. e.g.
Those two URIs are equivalent:

http://example.org/my-folder/my-document
HTTP://EXAMPLE.ORG/my-folder/my-document

While this one is not equivalent to the two URIs above:

http://example.org/My-Folder/my-document

To avoid confusion we always use lowercase letters in URIs.

37.2 Naming Convention for attribute names in URIs

Rule: Use hyphens (spinal case) to improve readability of URIs.
Do:

/users/noam/reset-password

Don’t:

/users/noam/resetPassword
/users/noam/ResetPassword
/users/noam/reset_password
Reason:

Spinal case is better to read and safer to use than camelCase (URLs are case sensitive (RFC3986)). Plone uses spinal case for URL creation (title “My page” becomes “my-page”) and mixed naming conventions in URLs would be confusing (e.g. “/my-folder/@send_url_to_user”). Google recommends spinal-case in URLs for better SEO (https://support.google.com/webmasters/answer/76329).

Discussion:

https://github.com/plone/plone.restapi/issues/194

### 37.3 Naming Convention for attribute names in response body

Rule: Use snake_case to reflect Python best practices.

Do:

```python
{
    translation_of: ...
}
```

Don’t:

```python
{
    translationOf: ...,
    TranslationOf: ...,
}
```

Reason:

We map over Python attributes 1:1 no matter if they are snake case (modern Python/Plone, Dexterity) of lowerCamelCase (Zope 2, Archetypes).

### 37.4 Versioning

Versioning APIs does make a lot of sense for public API services. Especially if an API provider needs to ship different versions of the API at the same time. Though, Plone already has a way to version packages and it currently does not make sense for us to expose that information via the API. We will always just ship one version of the API at a time and we are usually in full control over the backend and the frontend.
Note: This is only available on Plone 5.

Since Plone 5 the product `plone.app.multilingual` is included in the base Plone installation although it is not enabled by default.

Multilingualism in Plone not only allows the managers of the site to configure the site interface texts to be in one language or another (such as the configuration menus, error messages, information messages or other static text) but also to configure Plone to handle multilingual content. To achieve that it provides the user interface for managing content translations.

You can get additional information about the multilingual capabilities of Plone in the documentation.

In connection with that capabilities, `plone.restapi` provides a `@translations` endpoint to handle the translation information of the content objects.

Once we have installed `plone.app.multilingual` and enabled more than one language we can link two content-items of different languages to be the translation of each other issuing a `POST` query to the `@translations` endpoint including the `id` of the content which should be linked to. The `id` of the content must be a full URL of the content object: http

```bash
POST /plone/en/test-document/@translations HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
    "id": "http://localhost:55001/plone/es/test-document"
}
```

curl

```bash
```
httpie

```bash
echo '{
   "id": "http://localhost:55001/plone/es/test-document"
```

**python-requests**

```python
```

**Note:** “id” is a required field and needs to point to an existing content on the site.

The API will return a **201 Created** response if the linking was successful.

```
HTTP/1.1 201 Created
Content-Type: application/json

{}
```

We can also use the object’s path to link the translation instead of the full URL: http

```
POST /plone/en/test-document/@translations HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
   "id": "/es/test-document"
}
```

curl

```bash
   application/json' -H 'Content-Type: application/json' --data-raw '{"id": "/es/test-
   document"}' --user admin:secret
```

httpie

```bash
echo '{
   "id": "/es/test-document"
```

**python-requests**

```python
requests.post('http://nohost/plone/en/test-document/@translations', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'id': '/es/test-document'}, auth=('admin', 'secret'))
```

```
HTTP/1.1 201 Created
Content-Type: application/json
```

(continues on next page)
{}

We can also use the object’s UID to link the translation: http

```
POST /plone/en/test-document/@translations HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json
{
    "id": "SomeUUID000000000000000000000003"
}
```

curl

```
```

httpie

```
echo '{
    "id": "SomeUUID000000000000000000000003"
```

python-requests

```
requests.post('http://nohost/plone/en/test-document/@translations', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'id': 'SomeUUID000000000000000000000003'}, auth=('admin', 'secret'))
```

```
HTTP/1.1 201 Created
Content-Type: application/json
{}
```

After linking the contents we can get the list of the translations of that content item by issuing a GET request on the @translations endpoint of that content item.: http

```
GET /plone/en/test-document/@translations HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
```

httpie

```
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "items": [
      {
         "language": "es"
      }
   ]
}
```

To unlink the content, issue a DELETE request on the @translations endpoint of the content item and provide the language code you want to unlink.

```
DELETE /plone/en/test-document/@translations HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json

{
   "language": "es"
}
```

curl


httpie

echo '{
   "language": "es"

requests.delete('http://nohost/plone/en/test-document/@translations', headers={'Accept': 'application/json', 'Content-Type': 'application/json'}, json={'language': 'es'}, auth=('admin', 'secret'))

**Note:** “language” is a required field.
plone.restapi Documentation, Release 1.0a1

38.1 Creating a translation from an existing content
The POST content endpoint to a folder is capable also of linking this new content with an exising translation using
two parameters: translationOf and language. http
POST /plone/de HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
Content-Type: application/json
{
"@type": "Document",
"id": "mydocument",
"language": "de",
"title": "My German Document",
"translation_of": "SomeUUID000000000000000000000003"
}

curl
˓→"language": "de", "title": "My German Document", "translation_of":
˓→"SomeUUID000000000000000000000003"}' --user admin:secret

httpie
echo '{
"@type": "Document",
"id": "mydocument",
"language": "de",
"title": "My German Document",
"translation_of": "SomeUUID000000000000000000000003"
}' | http POST http://nohost/plone/de Accept:application/json Content˓→Type:application/json -a admin:secret

python-requests
requests.post('http://nohost/plone/de', headers={'Accept': 'application/json',
˓→'Content-Type': 'application/json'}, json={'@type': 'Document', 'id': 'mydocument',
˓→'language': 'de', 'title': 'My German Document', 'translation_of':
˓→'SomeUUID000000000000000000000003'}, auth=('admin', 'secret'))
HTTP/1.1 201 Created
Content-Type: application/json
Location: http://localhost:55001/plone/de/mydocument
{
"@components": {
"actions": {
"@id": "http://localhost:55001/plone/de/mydocument/@actions"
},
"breadcrumbs": {
"@id": "http://localhost:55001/plone/de/mydocument/@breadcrumbs"
},
"contextnavigation": {
"@id": "http://localhost:55001/plone/de/mydocument/@contextnavigation"
(continues on next page)

38.1. Creating a translation from an existing content

309


"navigation": {
  "@id": "http://localhost:55001/plone/de/mydocument/@navigation"
},
"translations": {
  "@id": "http://localhost:55001/plone/de/mydocument/@translations"
},
"types": {
  "@id": "http://localhost:55001/plone/de/mydocument/@types"
},
"workflow": {
  "@id": "http://localhost:55001/plone/de/mydocument/@workflow"
}
"@id": "http://localhost:55001/plone/de/mydocument",
"@type": "Document",
"UID": "SomeUUID00000000000000000000000000000005",
"allow_discussion": false,
"changeNote": "",
"contributors": [],
"created": "1995-07-31T13:45:00",
"creators": [
  "admin"
],
"description": "",
"effective": null,
"exclude_from NAV": false,
"expires": null,
"id": "mydocument",
"is_folderish": false,
"language": {
  "title": "Deutsch",
  "token": "de"
},
"layout": "document_view",
"modified": "1995-07-31T13:45:00",
"next_item": {},
"parent": {
  "@id": "http://localhost:55001/plone/de",
  "@type": "LRF",
  "description": "",
  "review_state": "published",
  "title": "Deutsch"
},
"previous_item": {
  "@id": "http://localhost:55001/plone/de/assets",
  "@type": "LIF",
  "description": "",
  "title": "Assets"
},
"relatedItems": [],
"review_state": "private",
"rights": "",
"subjects": [],
"table_of_contents": null,
"text": null,
"title": "My German Document"}
"version": "current",
"versioning_enabled": true,
"working_copy": null,
"working_copy_of": null
}

38.2 Get location in the tree for new translations

When you create a translation in Plone, there are policies in place for finding a suitable placement for it. This endpoint returns the proper placement for the newly going to be created translation.

http
GET /plone/es/test-document/@translation-locator?target_language=de HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0

curl


httpie

   --Accept:application/json -a admin:secret

python-requests


HTTP/1.1 200 OK
Content-Type: application/json

{   "@id": "http://localhost:55001/plone/de"
}

38.3 Expansion

This endpoint can be used with the Expansion mechanism which allows to get additional information about a content item in one query, avoiding unnecessary requests.

If a simple GET request is done on the content item, a new entry will be shown on the @components entry with the URL of the @translations endpoint:
39.1 Send Mail to Arbitrary Addresses

To send an email to an arbitrary e-mail address, send a POST request to the /@email-send endpoint that is available on the site root:

```
POST http://localhost:8080/Plone/@email-send
Accept: application/json
Content-Type: application/json

{
    'name': 'John Doe',
    'from': 'john@doe.com',
    'to': 'jane@doe.com',
    'subject': 'Hello!',
    'message': 'Just want to say hi.'
}
```

This endpoint is controlled via the Use mailhost services permission, the default one in Zope.

The ‘to’, ‘from’ and ‘message’ fields are required. The ‘subject’ and ‘name’ fields are optional.

The server will respond with status **204 No Content** when the email has been sent successfully:
i18n: internationalization of screen messages

Plone already provides user-interface translations using the `plone.app.locales` packages.

In `plone.restapi` we also use those translations where the end user needs to have those translated strings, this way the front-end work is easier, because you directly get from the server everything you need, instead of needing to query yet another endpoint to get the translations.

To do so, `plone.restapi` relies on Plone’s language-negotiation configuration and lets Plone do the work of deciding the language in which the messages should be shown.

For the content of a multilingual site built using `plone.app.multilingual` this is an easy task: Plone is configured to show in the language of the content-object, so there is no need to ask anything to the REST API.

Nevertheless, when you want to query the Plone Site object of a multilingual site, or any other endpoint in a plain Plone site with multiple languages configured, you need to query the REST API which language do you want to have the messages on, otherwise you will get the messages on the default language configured in Plone.

To achieve that, the REST API requires to use the `Accept-Language` HTTP header passing as the value the code of the required language.

You will also need to configure Plone to use the browser request language negotiation. To do so, you need to go the Plone Control Panel, go to the Language Control Panel, open the Negotiation configuration tab and select “Use browser language request negotiation” option.

Using this option we can get the content-type titles translated: http

```
GET /plone/@types HTTP/1.1
Accept: application/json
Accept-Language: es
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
```

httpie
  → admin:secret

python-requests

requests.get('http://nohost/plone/@types', headers={'Accept': 'application/json',
  → 'Accept-Language': 'es'}, auth=('admin', 'secret'))

And the response:

HTTP/1.1 200 OK
Content-Type: application/json

[
  
  "@id": "http://localhost:55001/plone/@types/File",
  "addable": true,
  "title": "Archivo"
  
  ],

  
  "@id": "http://localhost:55001/plone/@types/Folder",
  "addable": true,
  "title": "Carpeta"
  
  ],

  
  "@id": "http://localhost:55001/plone/@types/Collection",
  "addable": true,
  "title": "Colección"
  
  ],

  
  "@id": "http://localhost:55001/plone/@types/DXTestDocument",
  "addable": true,
  "title": "DX Test Document"
  
  ],

  
  "@id": "http://localhost:55001/plone/@types/Link",
  "addable": true,
  "title": "Enlace"
  
  ],

  
  "@id": "http://localhost:55001/plone/@types/Event",
  "addable": true,
  "title": "Evento"
  
  ],

  
  "@id": "http://localhost:55001/plone/@types/Image",
  "addable": true,
  "title": "Imagen"
  
  ],

  
  "@id": "http://localhost:55001/plone/@types/News Item",
  "addable": true,
  "title": "Noticia"
  
  ],

  
  "@id": "http://localhost:55001/plone/@types/Document",
  "addable": true,

(continues on next page)
All the field titles and descriptions will also be translated. For instance for the Folder content type:

GET /plone/@types/Folder HTTP/1.1
Accept: application/json
Accept-Language: es
Authorization: Basic YWRtaW46c2VjcmV0

curl


httpie


requests

requests.get('http://nohost/plone/@types/Folder', headers={"Accept": 'application/json'}, auth=('admin', 'secret'))

And the response:

HTTP/1.1 200 OK
Content-Type: application/json+schema

{
    "fieldsets": [
        {
            "behavior": "plone",
            "fields": [
                "title",
                "description"
            ],
            "id": "default",
            "title": "Por defecto"
        },
        {
            "behavior": "plone",
            "fields": [
                "subjects",
                "language",
                "relatedItems"
            ],
            "id": "categorization",
            "title": "Categorización"
        },
        {
            "behavior": "plone",
            "fields": [
                "effective"
            ],
            "id": "fecha",
            "title": "Fecha"
        }
    ]
}
"expires",
],
"id": "dates",
"title": "Fechas"
},
{
"behavior": "plone",
"fields": [
"creators",
"contributors",
"rights"
],
"id": "ownership",
"title": "Propiedad"
},
{
"behavior": "plone",
"fields": [
"allow_discussion",
"exclude_from_nav",
"id",
"nextPreviousEnabled"
],
"id": "settings",
"title": "Configuración"
}
],
"layouts": [
"album_view",
"event_listing",
"full_view",
"listing_view",
"summary_view",
"tabular_view"
],
"properties": {
"allow_discussion": {
"behavior": "plone.allowdiscussion",
"choices": [
["True",
"Sí"
],
["False",
"No"
]
],
"description": "Permitir comentarios para este tipo de contenido",
"enum": [
"True",
"False"
],
"enumNames": [
"Sí",
"No"
]
}
"factory": "Choice",
"title": "Permitir comentarios",
"type": "string",
"vocabulary": {
  "@id": "http://localhost:55001/plone/@sources/allow_discussion"
}
},
"contributors": {
  "additionalItems": true,
  "behavior": "plone.dublincore",
  "description": "Los nombres de las personas que han contribuido a este elemento. Cada colaborador debe estar en una línea independiente.",
  "factory": "Tuple",
  "items": {
    "description": "",
    "factory": "Text line (String)",
    "title": "",
    "type": "string"
  },
  "title": "Colaboradores",
  "type": "array",
  "uniqueItems": true,
  "widgetOptions": {
    "vocabulary": {
      "@id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabularies.Users"
    }
  }
},
"creators": {
  "additionalItems": true,
  "behavior": "plone.dublincore",
  "description": "Personas responsables de la creación del contenido de este elemento. Por favor, introduzca una lista de nombres de usuario, uno por línea. El autor principal debe ser el primero.",
  "factory": "Tuple",
  "items": {
    "description": "",
    "factory": "Text line (String)",
    "title": "",
    "type": "string"
  },
  "title": "Creadores",
  "type": "array",
  "uniqueItems": true,
  "widgetOptions": {
    "vocabulary": {
      "@id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabularies.Users"
    }
  }
},
"description": {
  "behavior": "plone.dublincore",
  "description": "Usado en listados de elementos y resultados de búsquedas.",
  "factory": "Text",
  "title": "Descripción"
}
"type": "string",
"widget": "textarea"
},
"effective": {
"behavior": "plone.dublincore",
"description": "La fecha en la que el documento será publicado. Si no selecciona ninguna fecha, el documento será publicado inmediatamente.",
"factory": "Date/Time",
"title": "Fecha de Publicación",
"type": "string",
"widget": "datetime"
},
"exclude_from_nav": {
"behavior": "plone.excludefromnavigation",
"default": false,
"description": "Si está marcado, este elemento no aparecerá en el árbol de navegación",
"factory": "Yes/No",
"title": "Excluir de la navegación",
"type": "boolean"
},
"expires": {
"behavior": "plone.dublincore",
"description": "La fecha en la que expira el documento. Esto hará automáticamente el documento invisible a otros a una fecha dada. Si no elige ninguna fecha, nunca expirará.",
"factory": "Date/Time",
"title": "Fecha de Terminación",
"type": "string",
"widget": "datetime"
},
"id": {
"behavior": "plone.shortname",
"description": "Este nombre se mostrará en la URL.",
"factory": "Text line (String)",
"title": "Nombre corto",
"type": "string"
},
"language": {
"behavior": "plone.dublincore",
"default": "en",
"description": "",
"factory": "Choice",
"title": "Idioma",
"type": "string",
"vocabulary": {
"@id": "http://localhost:55001/plone/@vocabularies/plone.app.vocabulariesSupportedContentLanguages"
}
},
"nextPreviousEnabled": {
"behavior": "plone.nextprevioustoggle",
"default": false,
"description": "Esto habilita el widget siguiente/prximo en los elementos contenidos en esta carpeta.",
"factory": "Yes/No",
"title": "Habilitar la navegación siguiente/anterior",
"type": "boolean"
}
"type": "boolean",
"relatedItems": {
  "additionalItems": true,
  "behavior": "plone.relateditems",
  "default": [],
  "description": ",",
  "factory": "Relation List",
  "items": {
    "description": ",",
    "factory": "Relation Choice",
    "title": "Related",
    "type": "string",
    "vocabulary": {
♭−Catalog"
    }
  },
  "title": "Contenido relacionado",
  "type": "array",
  "uniqueItems": true,
  "widgetOptions": {
    "pattern_options": {
      "recentlyUsed": true
    },
    "vocabulary": {
♭−Catalog"
    }
  }
},
"rights": {
  "behavior": "plone.dublincore",
  "description": "Declaramos los derechos sobre este elemento.",
  "factory": "Text",
  "title": "Derechos de Autor",
  "type": "string",
  "widget": "textarea"
},
"subjects": {
  "additionalItems": true,
  "behavior": "plone.dublincore",
  "description": "Las etiquetas suelen utilizarse para la organización del contenido.",
  "factory": "Tuple",
  "items": {
    "description": ",",
    "factory": "Text line (String)",
    "title": ",",
    "type": "string"
  },
  "title": "Etiquetas",
  "type": "array",
  "uniqueItems": true,
  "widgetOptions": {
    "vocabulary": {

In a given object, the workflow state and actions will be translated too: http

```
GET /plone/front-page/@workflow HTTP/1.1
Accept: application/json
Accept-Language: es
Authorization: Basic YWRtaW46c2VjcmV0
```

curl


httpie

```
```

python-requests

```
requests.get('http://nohost/plone/front-page/@workflow', headers={'Accept': 'application/json', 'Accept-Language': 'es'}, auth=('admin', 'secret'))
```

And the response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "@id": "http://localhost:55001/plone/front-page/@workflow",
    "history": [
        {
            "action": null,
            "actor": "test_user_1",
            "comments": ",
            "review_state": "private",
            "time": "1995-07-31T17:30:00",
```

(continues on next page)
"title": "Privado"
},
"state": {
"id": "private",
"title": "Privado"
},
"transitions": [
{
"@id": "http://localhost:55001/plone/front-page/@workflow/publish",
"title": "Publicar"
},
{
"@id": "http://localhost:55001/plone/front-page/@workflow/submit",
"title": "Enviar para publicación"
}
]
}

The same happens in the @history endpoint, all the relevant messages, will be shown translated: http

GET /plone/front-page/@history HTTP/1.1
Accept: application/json
Accept-Language: es
Authorization: Basic YWRtaW46c2VjcmV0
curl
httpie
python-requests
requests.get('http://nohost/plone/front-page/@history', headers={"Accept": "application/json", 'Accept-Language': 'es'}, auth=("admin", 'secret'))

And the response:

HTTP/1.1 200 OK
Content-Type: application/json

[ {
  "@id": "http://localhost:55001/plone/front-page/@history/0",
  "action": "Editado",
  "actor": {
    "@id": "http://localhost:55001/plone/users/test-user",
    "fullname": "test-user",
    "id": "test-user",
    "username": null
  }
} ]
"comments": "Versión inicial",
"may_revert": true,
"time": "1995-07-31T17:30:00",
"transition_title": "Editado",
"type": "versioning",
"version": 0
},
{
"action": "Crear",
"actor": {
"@id": "http://localhost:55001/plone/@users/test_user_1_",
"fullname": "test_user_1_",
"id": "test_user_1_",
"username": null
},
"comments": 
"review_state": "private",
"state_title": "Privado",
"time": "1995-07-31T18:30:00",
"transition_title": "Crear",
"type": "workflow"
Email Notification

41.1 Contact Site Owner aka Contact Form

Plone allows the user to contact the site owner via a form on the website. This makes sure the site owner does not have to expose their email addresses publicly and at the same time allow the users to reach out to the site owners.

To send an email notification to the site owner, send a POST request to the `/@email-notification` endpoint that is available on the site root:

```
POST http://localhost:8080/Plone/@email-notification
Accept: application/json
Content-Type: application/json

{
    'name': 'John Doe',
    'from': 'john@doe.com',
    'subject': 'Hello! ',
    'message': 'Just want to say hi.'
}
```

The ‘from’ and ‘message’ fields are required. The ‘subject’ and ‘name’ fields are optional.

The server will respond with status **204 No Content** when the email has been sent successfully:

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

41.2 Contact Portal Users

**Note:** This endpoint is NOT implemented yet.
To send an email notification to another user of the portal, send a POST request to the `/@email-notification` endpoint on a particular user (e.g. the admin user):

```
POST http://localhost:8080/Plone/@users/admin/@email-notification
Accept: application/json
Content-Type: application/json

{
    'name': 'John Doe',
    'from': 'john@doe.com',
    'subject': 'Hello!',
    'message': 'Just want to say hi.'
}
```

**Note:** When using “email as login”, we strongly recommend to also enable the “Use UUID user ids” setting in the security control panel, to obfuscate the email in the user endpoint URL. Otherwise the `@users` endpoint will expose the email addresses of all your users.
This upgrade guide lists all breaking changes in plone.restapi and explains the necessary steps that are needed to upgrade to the latest version.

### 42.1 Upgrading to plone.restapi 7.x

Navigation endpoint has been refactored, and now its behavior is consistent regarding the `items` attribute. Now the `items` attribute is present even if the element of the tree does not have children elements. This might affect some login in JavaScript specially, if the condition is checking for the existence of the attribute and expect to be `undefined`, since this change it will be an empty array.

### 42.2 Upgrading to plone.restapi 6.x

plone.restapi 6.0.0 removes the IAPIRequest marker interface (https://github.com/plone/plone.restapi/pull/819).

It also ships with a fix that prevents converting bytestring ids to unicode ids when reordering on Python 2 (https://github.com/plone/plone.restapi/issues/827).

All versions before plone.restapi 6.0.0 are potentially affected by this issue.

You may be affected by this issue and should run the fix if:

- You used the PATCH “ordering” functionality of plone.restapi
- Were using Python 2 at that point
- Are seeing issues with objectIds() returning mixed string types

If you need to fix object ids you can do one of the following:

- Use the browser-view `@@plone-restapi-upgrade-fix-ordering` as a “Manager” to fix all folderish content types in your Plone site.
Run the helper function `ensure_child_ordering_object_ids_are_native_strings` from `plone.restapi.upgrades.ordering` for all affected objects. You could do this in a custom upgrade-step implemented in your policy.

We expect that most content won’t actually be affected. See https://github.com/plone/plone.restapi/issues/827 for more details.

### 42.3 Upgrading to plone.restapi 5.x

plone.restapi 5.0.0 introduces the following breaking change:

- Rename tiles behavior and fields to blocks, migration step. [timo, sneridagh] (#821)

The “tiles” field has been renamed to “blocks” and the “tiles_layout” field to “blocks_layout”. This changes the response format from:

```json
{
   "@id": "http://localhost:55001/plone/my-document",
   ...
   "tiles_layout": [
      "#title-1",
      "#description-1",
      "#image-1"
   ],
   "tiles": {
      ...
   }
}
```

to:

```json
{
   "@id": "http://localhost:55001/plone/my-document",
   ...
   "blocks_layout": [
      "#title-1",
      "#description-1",
      "#image-1"
   ],
   "blocks": {
      ...
   }
}
```

This change affects the GET, PATCH and POST formats. Though, it should only affect you if you use Volto.

### 42.4 Upgrading to plone.restapi 4.x

plone.restapi 4.0.0 introduces the following breaking changes:

1) Fields with vocabularies now return the `token` and `title` instead of the stored value.

2) Choice and list fields return a hyperlink to a vocabulary instead of `choices`, `enum`, and `enumNames`.

3) Serialize widget parameters into a `widgetOptions` object instead of adding them to the top level of the schema property.
4) The vocabularies endpoint does no longer returns an @id for terms, the results are batched, and terms are now listed as items instead of terms to match other batched responses.

### 42.4.1 Serialization and Deserialization of fields with vocabularies

The serialization of fields with vocabularies (e.g. Choice) now return the token and the title of the vocabulary term instead of the stored value. This is allows displaying the term (title) without additionally querying the vocabulary. However it’s necessary to adopt existing client implementations.

The date and time controlpanel previously returned a number for the first_weekday property:

```json
{
  "@id": "http://localhost:55001/plone/@controlpanels/date-and-time",
  "data": {
    ...
    "first_weekday": 0,
    ...
  }
  ...
}
```

Now it returns an object with a token and a title:

```json
{
  "@id": "http://localhost:55001/plone/@controlpanels/date-and-time",
  "data": {
    ...
    "first_weekday": {
      "title": "Monday",
      "token": "0"
    },
    ...
  }
  ...
}
```

Deserialization accepts objects that contain a token, but also just the token or the value. However it’s highly recommended to always use the token as vocabulary terms may contain values that are not JSON serializable.

### 42.4.2 Choice and List fields return link to vocabulary instead of the values

Choice and List fields using named vocabularies are now serialized with a vocabulary property giving the URL of the @vocabularies endpoint for the vocabulary instead of including choices, enum and enumNames inline.

Old Response:

```
"choices": [
  [
    "de",
    "Deutsch"
  ],
  [
    "en",
    "English"
  ]
```

(continues on next page)
New response:

```json
"vocabulary": {
  →vocabularies.CaptchaVocabulary"
},
```

### 42.4.3 Serialize widget parameters into a `widgetOptions` object

Serialize widget parameters into a `widgetOptions` object instead of adding them to the top level of the schema property.

Old response:

```
"vocabulary": "plone.app.vocabularies.Users"
```

New response:

```
"widgetOptions": {
  "pattern_options": {
    "recentlyUsed": true
  },
  →vocabularies.Users" }
}
```

### 42.4.4 Example: Vocabularies Subjects Field

The `subjects` field is now serialized as an array of string items using the `plone.app.vocabularies` Keywords vocabulary.

Old response:

```
"subjects": {
  "choices": [...],
  "enum": [...],
  "enumNames": [...],
}
```

New response:

```
"type": "string"
```
"additionalItems": true,
"type": "array",
"uniqueItems": true,
"widgetOptions": {
   "vocabulary": {
       "Keywords"
   },
},
"items": {
   "description": "",
   "title": "",
   "type": "string"
},

42.4.5 Example: Available Time Zones Field (vocabulary in items)

Old response:

"available_timezones": {
   "additionalItems": true,
   "default": [],
   "description": "The timezones, which should be available for the portal. Can be set
   for users and events",
   "items": {
      "choices": [
         ["Africa/Abidjan",
          "Africa/Abidjan"
         ],
         ["Africa/Accra",
          "Africa/Accra"
         ],
         ...
      ],
      "enum": [
         ...
      ],
      "enumNames": [
         ...
      ],
      title: "Available timezones",
      type: "array",
      uniqueItems: true,
   }
}

New response:

"available_timezones": {
   "additionalItems": true,
   "default": [],
   "description": "The timezones, which should be available for the portal. Can be set
   for users and events",
   "items": {
      "description": "",
      (continues on next page)
"title": "",  
"type": "string",  
"vocabulary": {  
  →Timezones"
},  
"title": "Available timezones",  
"type": "array",  
"uniqueItems": true
,

42.4.6 Example: Weekday Field (vocabulary in main property)

Old response:

"first_weekday": {  
  "choices": [  
    [  
      "0",  
      "Monday"
    ],  
    [  
      "1",  
      "Tuesday"
    ],  
    [  
      "2",  
      "Wednesday"
    ],  
    [  
      "3",  
      "Thursday"
    ],  
    [  
      "4",  
      "Friday"
    ],  
    [  
      "5",  
      "Saturday"
    ],  
    [  
      "6",  
      "Sunday"
    ]
  ],  
  "description": "First day in the week.",  
  "enum": [  
    "0",  
    "1",  
    "2",  
    "3",  
    "4",  
    "5",
(continues on next page)
"first_weekday": {
    "description": "First day in the week."
    "title": "First weekday",
    "type": "string",
    "vocabulary": {
        "@id": "http://localhost:8080/Plone/@vocabularies/plone.app.vocabularies.Weekdays"
    }
},

42.4.7 Vocabularies Endpoint

The vocabularies endpoint does no longer returns an @id for terms.

The results are batched, and terms are now listed as items instead of terms to match other batched responses.

Batch size is 25 by default but can be overridden using the b_size parameter.

Old response:

```
{
    →ReallyUserFriendlyTypes",
    "terms": [
        {
    →ReallyUserFriendlyTypes/Collection",
            "title": "Collection",
            "token": "Collection"
        },
        ...
    ]
}
```

New response:

```
{
    →ReallyUserFriendlyTypes",
    "items": [
```
42.5 Upgrading to plone.restapi 3.x

42.5.1 Image scales

Image download URLs and image scale URLs are created using the UID based url formats. This allows Plone to create different URLs when the image changes and thus ensuring caches are updated.

Old Response:

```
{
    "icon": {
        "download": "http://localhost:55001/plone/image/@@images/image/icon",
        "height": 32,
        "width": 24
    },
    "large": {
        "download": "http://localhost:55001/plone/image/@@images/image/large",
        "height": 768,
        "width": 576
    },
    ...
}
```

New Response:

```
{
    "icon": {
        "download": "http://localhost:55001/plone/image/@@images/8eed3f80-5e1f-4115-85b8-650a10a6ca84.png",
        "height": 32,
        "width": 24
    },
    "large": {
        "height": 768,
        "width": 576
    },
    ...
}
```
42.5.2 @sharing endpoint

The available_roles property in the response to a GET request to the @sharing endpoint has changed: Instead of a flat list of strings, it now contains a list of dicts, with the role ID and their translated title:

Old Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "available_roles": [
    "Contributor",
    "Editor",
    "Reviewer",
    "Reader"
  ],
  "entries": [
    "...
  ],
  "inherit": true
}
```

New Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "available_roles": [
    {"id": "Contributor", "title": "Can add"},
    {"id": "Editor", "title": "Can edit"},
    {"id": "Reader", "title": "Can view"},
    {"id": "Reviewer", "title": "Can review"}
  ],
  "entries": [
    "...
  ],
  "inherit": true
}
```

42.5.3 Custom Content Deserializers

If you have implemented custom content deserializers, you have to handle the new create keyword in the __call__ method, which determines if deserialization is performed during object creation or while updating an
Deserializers should only fire an IObjectModifiedEvent event if an object has been updated. They should not fire it, when a new object has been created.

See Dexterity content deserializer for an example.

### 42.6 Upgrading to plone.restapi 2.x

plone.restapi 2.0.0 converts all datetime, DateTime and time to UTC before serializing. The translations endpoint becomes “expandable”, which introduces the following breaking changes.

#### 42.6.1 Translations

When using the @translations endpoint in plone.restapi 1.x, the endpoint returned a language key with the content object’s language and a translations key with all its translations.

Now, as the endpoint is expandable we want the endpoint to behave like the other expandable endpoints. As top level information we only include the name of the endpoint on the @id attribute and the actual translations of the content object in an attribute called items.

This means that now the JSON response to a GET request to the Translations endpoint does not include anymore the language of the actual content item and the translations in an attribute called items instead of translations.

Old response:

```json
HTTP/1.1 200 OK
Content-Type: application/json

{
    "language": "en",
    "translations": [
        {
            "language": "es"
        }
    ]
}
```

New response:

```json
HTTP/1.1 200 OK
Content-Type: application/json

{
    "items": [
        {
            "language": "es"
        }
    ]
}
```
42.7 Upgrading to plone.restapi 1.0b1

In plone.restapi 1.0b1 the ‘url’ attribute on the Navigation and Breadcrumbs endpoint was renamed to ‘@id’ to be consistent with other links/URLs used in plone.restapi.

The JSON response to a GET request to the Breadcrumbs endpoint changed from using the ‘url’ attribute for ‘items’:

```
HTTP/1.1 200 OK
Content-Type: application/json

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

to using the ‘@id’ for the URL of ‘items’:

```
HTTP/1.1 200 OK
Content-Type: application/json

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

The JSON response to a GET request to the Navigation endpoint changed from using the ‘url’ attribute for ‘items’:

```
HTTP/1.1 200 OK
Content-Type: application/json

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

to using the ‘@id’ for the URL of ‘items’:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "@id": "http://localhost:55001/plone/front-page/@navigation",
  "items": [
    {
      "@id": "http://localhost:55001/plone/front-page",
      "title": "Home",
      "url": "http://localhost:55001/plone",
    },
    {
      "@id": "http://localhost:55001/plone/front-page",
      "title": "Welcome to Plone",
      "url": "http://localhost:55001/plone/front-page"
    }
  ]
}
```
HTTP/1.1 200 OK
Content-Type: application/json

```json
{
    "@id": "http://localhost:55001/plone/front-page/@navigation",
    "items": [
        {
            "@id": "http://localhost:55001/plone",
            "title": "Home"
        },
        {
            "@id": "http://localhost:55001/plone/front-page",
            "title": "Welcome to Plone"
        }
    ]
}
```

The expansion mechanism is also affected by this change when Navigation or Breadcrumbs endpoints are expanded.

From using ‘url’ in the breadcrumb ‘items’:

```json
{
    "@components": {
        "breadcrumbs": {
            "@id": "http://localhost:55001/plone/front-page/@breadcrumbs",
            "items": [
                {
                    "title": "Welcome to Plone",
                    "url": "http://localhost:55001/plone/front-page"
                }
            ]
        },
        "navigation": {
            "@id": "http://localhost:55001/plone/front-page/@navigation",
            "items": [
                {
                    "title": "Home",
                    "url": "http://localhost:55001/plone",
                },
                {
                    "title": "Welcome to Plone",
                    "url": "http://localhost:55001/plone/front-page"
                }
            ]
        },
        ...
    }
}
```

to using ‘@id’ in the breadcrumb ‘items’:

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

(continues on next page)
Changelog:

- Rename 'url' attribute on navigation / breadcrumb to '@id'. [timo]

Pull Request:
  - https://github.com/plone/plone.restapi/pull/459

### 42.8 Upgrading to plone.restapi 1.0a25

plone.restapi 1.0a25 introduced three breaking changes:

- Remove @components navigation and breadcrumbs. Use top level @navigation and @breadcrumb endpoints instead. [timo]
- Remove “sharing” attributes from GET response. [timo,jaroel]
- Convert richtext using .output_relative_to. Direct conversion from RichText if no longer supported as we always need a context for the ITransformer. [jaroel]

#### 42.8.1 Remove @components endpoint

plone.restapi 1.0a25 removed the @components endpoint which used to provide a Navigation and a Breadcrumbs endpoint.

Instead of using “@components/navigation”:

[http://localhost:8080/Plone/@components/navigation](http://localhost:8080/Plone/@components/navigation)

Use just “@navigation”:

[http://localhost:8080/Plone/@navigation](http://localhost:8080/Plone/@navigation)

Instead of using “@components/breadcrumbs”:

[http://localhost:8080/Plone/@components/breadcrumbs](http://localhost:8080/Plone/@components/breadcrumbs)
Use just `@breadcrumbs`:

```
http://localhost:8080/Plone/@breadcrumbs
```

Changelog:

- Remove `@components` navigation and breadcrumbs. Use top level `@navigation and @breadcrumb` endpoints instead. [timo]

Pull Request:

- https://github.com/plone/plone.restapi/pull/425

### 42.8.2 Remove “sharing” attributes

The “sharing” attribute was removed from all content GET responses:

```
"sharing": {
    "@id": "http://localhost:55001/plone/collection/@sharing",
    "title": "Sharing"
},
```

Use the `Sharing` endpoint that can be expanded instead.

Changelog:

- Remove “sharing” attributes from GET response. [timo, jaroel]

Pull Request:

- https://github.com/plone/plone.restapi/commit/1b5e9e3a74df22e53b674849e27fa4b39b792b8c

### 42.8.3 Convert richtext using `.output_relative_to`

Using `.output_relative_to` in the

Changelog:

- Convert richtext using `.output_relative_to`. Direct conversion from RichText if no longer supported as we always need a context for the ITransformer. [jaroel]

Pull Request:

https://github.com/plone/plone.restapi/pull/428

### 42.9 Upgrading to plone.restapi 1.0a17

plone.restapi 1.0a17 changed the serialization of the rich-text “text” field for content objects from using ‘raw’ (a unicode string with the original input markup):
to using ‘output’ (a unicode object representing the transformed output):

```json
"text": {
  "content-type": "text/plain",
  "data": "<p>Lorem ipsum</p>",
  "encoding": "utf-8"
},
```

Changelog:

- Change RichText field value to use 'output' instead of 'raw' to fix inline paths. This fixes #302. [erral]

Pull Request:

https://github.com/plone/plone.restapi/pull/346
43.1 Generating documentation examples

This documentation includes examples of requests and responses (http, curl, httpie and python-requests). These examples are generated by the documentation tests in test_documentation.py. To generate a new example, add a new test case to test_documentation.py - for example test_documentation_search_fullobjects, and run the test:

```
./bin/test -t test_documentation_search_fullobjects
```

This generates the request and the response files in tests/http-examples/.

Include them in the documentation like this:

```
.. http example:: curl httpie python-requests
   :request: ../../../src/plone/restapi/tests/http-examples/search_fullobjects.req

.. literalinclude:: ../../../src/plone/restapi/tests/http-examples/search_fullobjects.resp
   :language: http
```

Build the sphinx docs locally to test the rendering by running ./bin/sphinxbuilder.

Make sure you add and commit the generated files in http-examples.
The @system endpoint exposes system information about the Plone backend.

Send a GET request to the @system endpoint: http

```
GET /plone/@system HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i http://nohost/plone/@system -H 'Accept: application/json' --user admin:secret
```

httpie

```
http http://nohost/plone/@system Accept:application/json -a admin:secret
```

python-requests

```
requests.get('http://nohost/plone/@system', headers={'Accept': 'application/json'},
              auth=('admin', 'secret'))
```

The response will contain the system information:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "@id": "http://localhost:55001/plone/@system",
    "cmf_version": "2.4.2",
    "debug_mode": "No",
    "pil_version": "6.2.1 (Pillow)",
    "plone_gs_metadata_version_file_system": "5208",
    "plone_gs_metadata_version_installed": "5208",
    "plone_version": "5.2.1",
}
```

(continues on next page)
"python_version": "3.7.7 (default, Mar 10 2020, 15:43:33) \n[Clang 11.0.0 (clang-
\n→1100.0.33.17)]",
  "zope_version": "4.1.3"
}

Note: The system endpoint is protected by the plone.app.controlpanel.Overview permission that requires the site-administrator or manager role.
The @database endpoint exposes system information about the Plone database (ZODB).

Send a GET request to the @database endpoint: http

```
GET /plone/@database HTTP/1.1
Accept: application/json
Authorization: Basic YWRtaW46c2VjcmV0
```

curl

```
curl -i http://nohost/plone/@database -H 'Accept: application/json' --user admin:secret
```

httpie

```
```

python-requests

```
requests.get('http://nohost/plone/@database', headers={'Accept': 'application/json'},
            auth=('admin', 'secret'))
```

The response will contain the database information:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "@id": "http://localhost:55001/plone/@database",
    "cache_detail_length": [ 
        {
            "connection": "<Connection at 11238e150>",
            "ngsize": 393,
            "size": 862
        }
    ]
}
```
Note: The system endpoint is protected by the `plone.app.controlpanel.Overview` permission that requires the site-administrator or manager role.
plone.restapi is a RESTful hypermedia API for Plone.
CHAPTER 47

Documentation

https://plonerestapi.readthedocs.org
A live demo of Plone 5 with the latest plone.restapi release is available at:
https://plonedemo.kitconcept.com

Example GET request on the portal root

```
```

Example POST request to create a new document

```
```

**Note:** You will need some kind of API browser application to explore the API. We recommend using *Postman.*
Install plone.restapi by adding it to your buildout:

```
[buildout]
...

eggs =
    plone.restapi
```

and then running `bin/buildout`
CONTRIBUTE

- **Issue Tracker:** https://github.com/plone/plone.restapi/issues
- **Source Code:** https://github.com/plone/plone.restapi
- **Documentation:** https://plonerestapi.readthedocs.io/en/latest
Examples

plone.restapi is used in production since the first alpha release. It can be seen in action at the following sites:

- Zeelandia GmbH & Co. KG: https://www.zeelandia.de (by kitconcept GmbH)
- VHS-Ehrenamtsportal: https://vhs-ehrenamtsportal.de (by kitconcept GmbH)
- German Physical Society: https://www.dpg-physik.de (by kitconcept GmbH)
- Universitat Politecnica de Catalunya: https://www.upc.edu (by kitconcept GmbH)
If you are having issues, please let us know via the issue tracker.

If you required professional support, here is a list of Plone solution providers that contributed significantly to plone.restapi in the past:

- kitconcept GmbH (Germany)
- 4teamwork (Switzerland)
- CodeSyntax (Spain)
The project is licensed under the GPLv2.

## 53.1 Appendix, Indices and tables

### 53.1.1 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.

53.1.2 Glossary

Accept Header Part of the Request that is responsible to define the expected type of data to be accepted by the client in the Response.

Authentication Method Access restriction provided by the connection chain to the server exposed to the client.

Authorization Header Part of the Request that is responsible for the authentication related to the right user or service to ask for a Response.

Basic Auth A simple Authentication Method referenced in the Authorization Header that needs to be provided by the server.

HTTP-Header

HTTP Header

Header The part of the communication of the client with the server that provides the initialisation of the communication of a Request.

HTTP-Request

HTTP Request

Request

Requests The initial action performed by a web client to communicate with a server. The Request is usually followed by a Response by the server, either synchronous or asynchronous (which is more complex to handle on the user side)

HTTP-Response

HTTP Response

Response Answer of or action by the server that is executed or send to the client after the Request is processed.

HTTP-Verb

HTTP Verb

Verb One of the basic actions that can be requested to be executed by the server (on an object) based on the Request.

Object URL The target object of the Request

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
GET (context)/@querysources/{field_name}?query={search_query}, 239
GET (context)/@sources/{field_name}, 238
GET (context)/@vocabularies, 230
GET (context)/@vocabularies/{vocab_name}, 235
GET (context)/@vocabularies/{vocab_name}?title={filter_query}, 236
GET (context)/@vocabularies/{vocab_name}?token={filter_query}, 236
Symbols
200 OK, 363
201 Created, 363
204 No Content, 363
2xx Success, 363
400 Bad Request, 363
401 Unauthorized, 363
403 Forbidden, 363
404 Not Found, 363
405 Method Not Allowed, 363
409 Conflict, 364
4xx Client Error, 364
500 Internal Server Error, 364
5xx Server Error, 364

A
Accept Header, 364
Authentication Method, 364
Authorization Header, 364

B
Basic Auth, 364

H
Header, 364
HTTP Header, 364
HTTP Request, 364
HTTP Response, 364
HTTP Verb, 364
HTTP-Header, 364
HTTP-Request, 364
HTTP-Response, 364
HTTP-Verb, 364

O
Object URL, 364

R
Request, 364

Requests, 364
Response, 364
REST, 364
V
Verb, 364
W
workflow, 364