diretto API v1 Documentation

Benjamin Erb, Christian Koch, Stefan Kaufmann

September 2010

License

Contents

1 The API

The API is the core of the whole diretto platform. It describes the external resource-oriented interfaces that clients and servers must implement. We tried to keep the API as simple as possbile while still laying the foundation for sophisticated usage. In this section, we first have a look at abstract composition of the API, followed by some technical details.

1.1 Overview

The API can be devided in to several interrelated resources:

1.1.1 User

Most of the API calls require authentication. Each participant needs a valid user session, not matter if he is a human user or some kind of bot that executes tasks automatically. The current user model is flat, which means that there are no user hierarchies and each user has the same rights than all others. There are minor expcetions when it comes to user profiles or deletion of content. A user is currently identified by his registration e-mail address and personal password. Users can also choose an alias /user name, which does not have to be unique and is primarily for displaying users.

1.1.2 Document

A document is a central entity in the diretto platform. Surprisingly, our documents do not represent distinct documents first-hand. Instead, they are abstract entities that group one or more concrete versions of the real document. This allows us the organize an original document and all its processed variants together. Also most of the meta data belong to the conceptual document rather than to distinct versions. A document belongs to a certain type of media, e.g. image.

1.1.3 Attachment

Each document has one more associated attachment, whereas the first attachment always represents the original version. An attachment is a concrete media document that can be stored as a file and has a distinct mime type. Users can add new attachments to a document. Added variants should be derived or processed versions of the original attachment and should not add unrelated content.

1.1.4 Tempo-Spatial Meta Data

Every document has some kind of origin that is bound to a specific point in time and a location. If the document itself has a temporal duration, the point in time always refers to the beginning of the creation. Our platform can handle inaccurate data, which means that also spatial areas or time ranges can be indicated. Temporal and spatial data are decoupled so that the Cartesian product of all potential temporal and spatial data lists all potential origins of the document. A user can either suggest a new moment and/or location for a document, or he can vote for existing entries.

1.1.5 Link

Interconnections among documents can be specified as links. Such links represent semantical meanings between documents, such as cause-and-effect connections. Like doucments, links can be enriched with tags or key-value pairs.

1.1.6 Tag

Each attachment can be annotated with a tag. The tags in our platform are weighted, thus users can up-vote or down-vote tags. Tags enable users to collaboratively build taxonomies for the documents.

1.1.7 Comment

A attachment can be commented by a user with a text message. This allows an exchange of thoughts between the different users or even between the author and the viewers.

1.1.8 Key-Value Pair

While all other types of user-generated content mostly focus on humans, key-value pairs provide a type also for machine-based consumption. Each user has its own namespace where he can store simple text values under a given key name mapped to an attachment. These values can be read by all other users. Thanks to the user-separated namespaces, users can use the same keys concurrently — values are still separated. Key-value pairs are helpful for annotating attachments with structural content that can be processed by bots or applications.

1.1.9 Voting

Most of the user-generated content can be rated with an up-vote/down-vote. With an up-vote the user agrees that the entry is useful, valid or advantageous. Down-votes can be used to filter out incorrect or invalid entries. Voting can be applied to attachments, links, tags, comments and tempo-spatial meta data.

1.2 Media Types

The platform is not limited to images or other distinct media types. Basically, every possible type of media document can be used. Therefore, we only store very generic meta data about each entry directly. Detailed and specific meta data, like the resolution of an image or the duration of a video can still be saved as a key-value entry to the document. In order to make handling of different media types simple, we setup a threefold classification. Different traits of media objects can be used to group certain media object classes. Each usable media type must now be listed with its mime type and attached to one of the classes. This allows

This classification allows an abstract view onto the different media objects. It allows clients to implement features based upon the media traits or media class profiles instead of concrete mime types, which makes the clients more versatile.

2 Guidelines for the API documentation

This documentation represents a technical documentation of our RESTful API. It is the primary source of information for implementing own client or server applications without using existing libraries. Each logical API action is mapped to a distinct HTTP request and described in detail. The following characteristics are used therefore:

URI The URI against the API call must be invoked. It may contain placeholders and parameters that are described below. In order to provide flexibility with future versions of the API, the APIs are versioned by adding a version parameter to the path. So for the current version, the string ”v1” must be appended to the server API root path, resulting in http://apipath/v1.

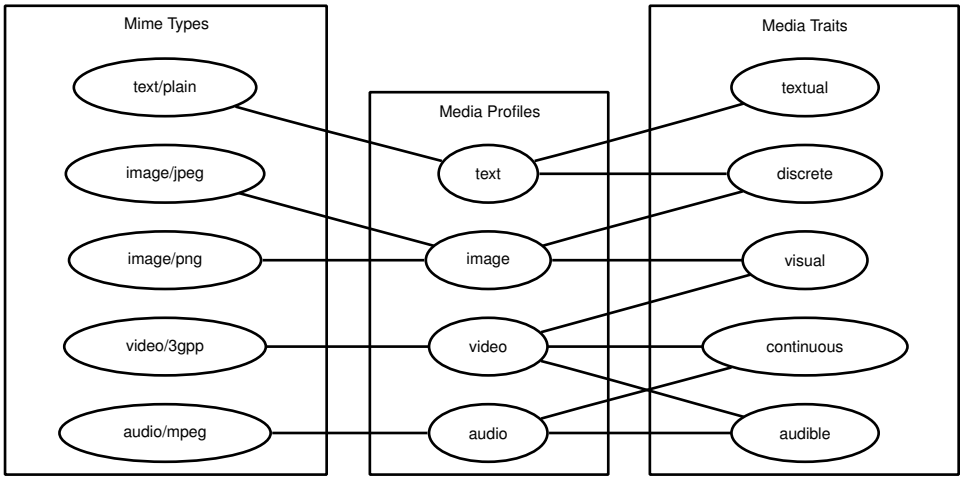


Figure 1: Threefold classification of different media types

HTTP Methods Lists the valid HTTP mehtod(s) that can be used to invoke this API call.

Parameters Lists all URI parameters and their allowed values.

Formats The valid data formats that can be used in this API call for entities.

Authentication Required Stats whether an authentication is required for this API call or not. Most of the API methods do need authentication.

Request Entity If a request contains an entity body, an exemplary entity is shown.

Responses Lists all HTTP response codes and their concrete meaning for this API call. Note that there are other repsonse codes that can be received on every request, like a `500` for an generic server-side error. The response codes listed in the an API call documentation are the codes specific for the current request.

Response Entity If a request contains an entity body, an exemplary entity is shown. If an error occurs, there is always a simple JSON object struct with the key `error` that contains a textual error message as its respective value.

3 Attachments

The following actions provide methods related to the document attachments and their meta data.

3.1 Get an attachment

Returns metadata of the attachment.

3.1.1 URI

`http://apipath/documents/` `doc-id` `/attachments/` `attachment-id`

3.1.2 HTTP Methods

`GET`

3.1.3 Parameters

doc-id	UUID of the document
attachment-id	UUID of the attachment

3.1.4 Formats

JSON

3.1.5 Authentication Required

true

3.1.6 Request Entity

n/a

3.1.7 Responses

200	Attachment metadata in body
202	Attachment known, but upload pending
404	Attachment not found

3.1.8 Response Entity

```
1 {
2    "attachmentId":"e5fba1fd-0717-4197-a40a-b2a5e079f715",
3    "docId":"e5fba1fd-0717-4197-a40a-b2a5e079f715",
4    "mimetype":"text/plain",
5    "size":1064,
6    "uploader":"foo",
7    "uploaded":"2010-07-17T18:54:11.947Z",
```

```
8    "location":"http://api.diretto.org:8000/e5fba1fd-0717-4197-a40a
         -b2a5e079f715/e5fba1fd-0717-4197-a40a-b2a5e079f715.txt"
9 }
```

3.2 Get the list of attachment IDs for a document

Returns metadata of the document

3.2.1 URI

`http://apipath/documents/` `doc-id` `/attachments`

3.2.2 HTTP Methods

`GET`

3.2.3 Parameters

doc-id	UUID of the document

3.2.4 Formats

JSON

3.2.5 Authentication Required

true

3.2.6 Request Entity

n/a

3.2.7 Responses

200	Attachment id list in body
404	Document not found

3.2.8 Response Entity

```
{
   "documentId":"e5fba1fd-0717-4197-a40a-b2a5e079f715",
   "attachmentIds":[
      "e5fba1fd-0717-4197-a40a-b2a5e079f715",
      "550e8400-e29b-11d4-a716-446655440000"
   ]
}
```

3.3 Forward from alias URI

Forwards the alias URI without document id to the correct URI including document.

3.3.1 URI

`http://apipath/attachments/` `attach-id`

3.3.2 HTTP Methods

`GET`

3.3.3 Parameters

attach-id	Attachment ID

3.3.4 Formats

n/a

3.3.5 Authentication Required

true

3.3.6 Request Entity

n/a

3.3.7 Responses

303 See Other + Location	Forward to correct URI
404 Not found	Attachment not found

3.3.8 Response Entity

n/a

3.4 List all attachements

Lists the id and mime type of all uploaded documents sorted by upload time. Results are split up into chunks of 100 documents. Navigation within the pagination is possible using cursor ids. These ids represent the first entry of a chunk of the previous/following result page. First/last page omits the previous/next key-value pair. A request without cursor will start with the first available value as cursor.

3.4.1 URI

- `http://apipath/attachments/` `ascending|descending` `/` =¿ will be 303'ed to `http://apipath/attachments/` `ascending|descending` `/cursor/` `firstcursor`
- `http://apipath/attachments/` `ascending|descending` `/cursor/` `documentid`

3.4.2 HTTP Methods

`GET`

3.4.3 Parameters

n/a

3.4.4 Formats

JSON

3.4.5 Authentication Required

true

3.4.6 Request Entity

n/a

3.4.7 Responses

200 Accepted	List delivered
303 See Other + Location including first cursor	Query without cursor
404 Not found	Cursor not found

3.4.8 Response Entity

```
{
  "attachments" : [
    {"id" : "3e64c371ba2b93f1c0fead369fe004ef", "mimetype" : "
        image/jpeg" },
    //98 other entries here...
    {"id" : "57ebb75106e99f6e7299fefe37de83b7", "mimetype" : "text
        /plain" }
  ],
  "next" : "acbd18db4cc2f85cedef654fccc4a4d8",
  "previous" : "37b51d194a7513e45b56f6524f2d51f2",
  "total" : 1234
}
```

4 Comments

4.1 Commentable Items

Comments can be added to the following items:

- Attachments
- base URI = `http://apipath/documents/` `doc-id` `/attachments/` `attachment-id`
- Links
- base URI = `http://apipath/links/` `link-id`

4.2 List comments by attachment

Returns a list of comments of a attachment.

4.2.1 URI

`http://apipath/documents/` `doc-id` `/attachments/` `attachment-id` `/comments`

4.2.2 HTTP Methods

`GET`

4.2.3 Parameters

doc-id	UUID of the document
attachment-id	UUID of the attachment

4.2.4 Formats

JSON

4.2.5 Authentication Required

true

4.2.6 Request Entity

n/a

4.2.7 Responses

200	List of comments
404	Document/Attachment not found

4.2.8 Response Entity


```
{
   "comments":[
      {
         "comment_id":"d4777513-7eb6-de3d-6c90-81cd8205c376",
         "doc_id":"e08eee5b-ba09-484a-b25d-1b7f2b7ae45c",
         "attachment_id":"e08eee5b-ba09-484a-b25d-1b7f2b7ae45c",
         "user":"foox",
         "comment":"blablabla",
         "created":"2010-09-02T14:22:13.744Z"
      },
      {
         "comment_id":"ffb4480c-5a9a-481a-821a-b4b3c0c917d0",
         "doc_id":"e08eee5b-ba09-484a-b25d-1b7f2b7ae45c",
         "attachment_id":"e08eee5b-ba09-484a-b25d-1b7f2b7ae45c",
         "user":"foox",
         "comment":"blablabla2",
         "created":"2010-09-02T14:22:13.844Z"
      }
   ],
   "total":2
}
```

4.3 Get a attachment comment by comment ID

Returns a comment by ID. Alternatively, you can query `http://apipath/comments/` `comment-id` and will be 303'ed to the right URL with document and attachment ID.

4.3.1 URI

`http://apipath/documents/` `doc-id` `/attachments/` `attachment-id` `/comments/` `comment-id`
`http://apipath/comments/` `comment-id` (see description above)

4.3.2 HTTP Methods

`GET`

4.3.3 Parameters

doc-id	UUID of the document
attachment-id	UUID of the attachment
comment-id	UUID of the comment

4.3.4 Formats

JSON

4.3.5 Authentication Required

true

4.3.6 Request Entity

n/a

4.3.7 Responses

200	Comment
404	Comment not found

4.3.8 Response Entity

```
1 {
2    "comment_id":"d4777513-7eb6-de3d-6c90-81cd8205c376",
3    "doc_id":"e08eee5b-ba09-484a-b25d-1b7f2b7ae45c",
4    "attachment_id":"e08eee5b-ba09-484a-b25d-1b7f2b7ae45c",
5    "created":"2010-09-02T14:22:13.744Z",
```

```
6     "type":"comment",
7     "user":"foox",
8     "comment":"blablabla"
9  }
```

4.4 Create a new comment for an attachment

Post a new comment to the given attachment.

4.4.1 URI

`http://apipath/documents/ doc-id /attachments/ attachment-id /comments`

4.4.2 HTTP Methods

`POST`

4.4.3 Parameters

doc-id	UUID of the document
attachment-id	UUID of the attachment

4.4.4 Formats

JSON

4.4.5 Authentication Required

true

4.4.6 Request Entity

```
1  {
2     "comment":"blablabla"
3  }
```

4.4.7 Responses

201	List of comments
400	Invalid comment data
404	Document/Attachment not found

4.4.8 Response Entity

n/a

4.5 List attachment comments by user

Returns a list of comments by user

4.5.1 URI

`http://apipath/users/user-id/comments`

4.5.2 HTTP Methods

`GET`

4.5.3 Parameters

user-id	The ID of the user

4.5.4 Formats

JSON

4.5.5 Authentication Required

true

4.5.6 Request Entity

n/a

4.5.7 Responses

200	List of comments
404	user not found

4.5.8 Response Entity

```
1 {
2    "comments":[
3       {
4          "comment_id":"d4777513-7eb6-de3d-6c90-81cd8205c376",
5          "doc_id":"e08eee5b-ba09-484a-b25d-1b7f2b7ae45c",
```

```
        "attachment_id":"e08eee5b-ba09-484a-b25d-1b7f2b7ae45c",
        "user":"foox",
        "comment":"blablabla",
        "created":"2010-09-02T14:22:13.744Z"
      }
   ],
   "total":1
}
```

4.6 List comments by link

Returns a list of comments of a link.

4.6.1 URI

`http://apipath/links/link-id/comments`

4.6.2 HTTP Methods

`GET`

4.6.3 Parameters

link-id	UUID of the link

4.6.4 Formats

JSON

4.6.5 Authentication Required

true

4.6.6 Request Entity

n/a

4.6.7 Responses

200	List of comments
404	Link not found

4.6.8 Response Entity

```
{
   "comments":[
      {
         "comment_id":"d4777513-7eb6-de3d-6c90-81cd8222b846",
         "link_id":"d4777513-7eb6-de3d-6c90-81cd82215410",
         "user":"foox",
         "comment":"bla",
         "created":"2010-09-12T19:17:46.344Z"
      },
      {
         "comment_id":"d4777513-7eb6-de3d-6c90-81cd8222c1e1",
         "link_id":"d4777513-7eb6-de3d-6c90-81cd82215410",
         "user":"foox",
         "comment":"bla",
         "created":"2010-09-12T19:18:02.334Z"
      }
   ],
   "total":2
}
```

4.7 Get a link comment by comment ID

Returns a comment by ID. Alternatively, you can query `http://apipath/comments/` `comment-id` and will be 303'ed to the right URL with document and attachment ID.

4.7.1 URI

`http://apipath/links/` `link-id` `/comments/` `comment-id` `http://apipath/comments/` `comment-id` (see description above)

4.7.2 HTTP Methods

`GET`

4.7.3 Parameters

link-id	UUID of the link
comment-id	UUID of the comment

4.7.4 Formats

JSON

4.7.5 Authentication Required

true

4.7.6 Request Entity

n/a

4.7.7 Responses

200	Comment
404	Comment not found

4.7.8 Response Entity

```
{
   "comment_id":"d4777513-7eb6-de3d-6c90-81cd8222c1e1",
   "link_id":"d4777513-7eb6-de3d-6c90-81cd82215410",
   "created":"2010-09-12T19:18:02.334Z",
   "type":"comment",
   "user":"foox",
   "comment":"bla"
}
```

4.8 Create a new comment for a link

Post a new comment to the given link.

4.8.1 URI

`http://apipath/links/link-id/comments`

4.8.2 HTTP Methods

`POST`

4.8.3 Parameters

link-id	UUID of the link

4.8.4 Formats

JSON

4.8.5 Authentication Required

true

4.8.6 Request Entity

```
{
   "comment":"blablabla"
}
```

4.8.7 Responses

201	List of comments
400	Invalid comment data
404	Link not found

4.8.8 Response Entity

n/a

5 Documents

The following actions provide methods related to the documents collected by the users and their meta data. [[TitleIndex(Components/API/v1/REST/Documents,format=group)]]

5.1 Get a document

Returns metadata of the document

5.1.1 URI

`http://apipath/documents/` `doc-id`

5.1.2 HTTP Methods

`GET`

5.1.3 Parameters

doc-id	UUID of the new document

5.1.4 Formats

JSON

5.1.5 Authentication Required

true

5.1.6 Request Entity

n/a

5.1.7 Responses

200	Document metadata in body
202	Document known, but upload pending
404	Document not found

5.1.8 Response Entity

```
{
  "id":"d16e103e-4fda-433d-9309-179487ecad4a",
  "mediatype":"text",
  "owner":"foo",
  "uploaded":"2010-07-17T18:54:11.939Z"
}
```

5.2 Metadata upload for a new document

Uploads metadata as an initial step for creating a new document.

5.2.1 URI

`http://apipath/documents/doc-id`

5.2.2 HTTP Methods

`PUT`

5.2.3 Parameters

doc-id	UUID of the new document

5.2.4 Formats

JSON

5.2.5 Authentication Required

true

5.2.6 Request Entity

`PUT /v1/documents/d16e103e-4fda-433d-9309-179487ecad4a`

```
{
   "document":{
      "date":{
         "before":"2010-06-29T19:15:51.765Z",
         "after":"2010-06-29T19:15:50.000Z"
      },
      "location":{
         "position":{
            "type":"Point",
            "coordinates":[
               100.0,
               0.0
            ]
         },
         "variance": 100
      }
   },
   "attachment":{
      "contentType":"image/jpeg",
      "contentLength":123
   }
}
```

5.2.7 Responses

202	Accepted + Entity
400	Invalid metadata
409	Document already exists
413	Document content content is to large
415	Document content type is not supported

5.2.8 Response Entity

```
{
   "upload":{
      "key":"00b19c9d7db4f1ae0824c62bda5a127588bde476",
      "location":"http://localhost:8000/d16e103e-4fda-433d
          -9309-179487ecad4a/d16e103e-4fda-433d-9309-179487ecad4a.
          jpg",
      "uri":"http://localhost:8000/d16e103e-4fda-433d-9309-179487
          ecad4a/d16e103e-4fda-433d-9309-179487ecad4a.jpg?key=00
          b19c9d7db4f1ae0824c62bda5a127588bde476"
   }
}
```

5.3 Confirm successful upload

As last step of an upload process, the client must confirm the successful upload by releasing document lock.

5.3.1 URI

`http://apipath/documents/` `doc-id` `/attachments/` `attach-id` `/lock/` `lock-id`

5.3.2 HTTP Methods

`DELETE`

5.3.3 Parameters

doc-id	UUID of the new document
attach-id	generated UUID of the attachment (equals doc-id in case of new attachments)
lock-id	Lock ID provided by the media node

5.3.4 Formats

n/a

5.3.5 Authentication Required

true

5.3.6 Request Entity

n/a

5.3.7 Responses

204	Upload completed
403	Invalid lock release (lock id does not match/wrong user)

5.3.8 Response Entity

n/a

5.4 List all documents

Lists the id and type of all uploaded documents sorted by upload time. Results are split up into chunks of 100 documents. Navigation within the pagination is possible using cursor ids. These ids represent the first entry of a chunk of the previous/following result page. First/last page omits the previous/next key-value pair. A request without cursor will start with the first available value as cursor.

5.4.1 URI

- `http://apipath/documents/` `ascending|descending` `/` =¿ will be 303'ed to `http://apipath/documents/` `ascending|descending` `/cursor/` `firstcursor`
- `http://apipath/documents/` `ascending|descending` `/cursor/` `documentid`

5.4.2 HTTP Methods

`GET`

5.4.3 Parameters

order	ascending or descending

5.4.4 Formats

JSON

5.4.5 Authentication Required

true

5.4.6 Request Entity

n/a

5.4.7 Responses

200 Accepted	List delivered
303 See Other + Location including first cursor	Query without cursor
404 Not found	Cursor not found

5.4.8 Response Entity

```
1  {
2    "documents" : [
3      {"id" : "3e64c371ba2b93f1c0fead369fe004ef", "type" : "image"
           },
4      //98 other entries here...
5      {"id" : "57ebb75106e99f6e7299fefe37de83b7", "type" : "video" }
6    ],
7    "next" : "acbd18db4cc2f85cedef654fccc4a4d8",
8    "previous" : "37b51d194a7513e45b56f6524f2d51f2",
9    "total" : 1234
10 }
```

5.5 List documents uploaded before or after a distinct point in time

This method allows to scroll through document lists by a given point in time as cursor. It either moves chronologically (descending) to newer documents added after the time (after semantics) or ascending to older ones (before semantics). If there is an exact match for the given time/date, this document will be used as resolved curosr, otherwise the next matching document in order will be. The time cursor must be a valid RFC 3339 timestamp like `2010-06-29T19:15:51.765Z`.

5.5.1 URI

- `http://apipath/documents/` `before|after` `/` `date` =¿ will be 303'ed to `http://apipath/documents/` `ascending|descending` `/cursor/` `firstcursor`

5.5.2 HTTP Methods

`GET`

5.5.3 Parameters

cursor type	before/after
date	Valid RFC 3339 timestamp used as split point

5.5.4 Formats

n/a

5.5.5 Authentication Required

true

5.5.6 Request Entity

n/a

5.5.7 Responses

204 Not found	No documents available
303 See Other + Location including first cursor	Query without cursor
404 Not found	Cursor not found

5.5.8 Response Entity

n/a

6 Index

This represents the root path of the diretto webservice.

6.1 Get service root

Returns information about the web service.

6.1.1 URI

`http://apipath`

6.1.2 HTTP Methods

`GET`

6.1.3 Parameters

n/a

6.1.4 Formats

JSON

6.1.5 Authentication Required

false

6.1.6 Request Entity

n/a

6.1.7 Responses

200 OK	Response entity containing the info

6.1.8 Response Entity

```
1  {
2     "version":"v1",
3     "deployment":{
4        "title":"diretto Project Team Development ",
5        "contact":"n/a",
6        "website":"http://www.diretto.org"
7     },
8     "service":{
9        "uri":"http://localhost:8001/"
10    },
11    "mediaserver":{
12       "uri":"http://localhost:8000/"
13    }
14 }
```

7 KeyValues

Users can add text-based key-value pairs to some entries. Each user has its own namespace, thus multiple users can use the same keys concurrently. A pair can only be created, modified and deleted by the user, but every other user has read-only access to it. So values should not be used for private or restricted information. Note that values must only be strings, so structured formats might need to be serialized first. A concrete service may also limit the amount of values per entry or the maximal size per value.

7.1 Types that can be labelled with a key-value pairs

- Attachments
- base URI = `http://apipath/documents/` `doc-id` `/attachments/` `attachment-id`
- Links
- base URI = `http://apipath/links/` `link-id`

7.2 Return a list of key/values of an attachment

List all pairs of the given attachment.

7.2.1 URI

`http://apipath/documents/` `doc-id` `/attachments/` `attachment-id` `/values`

7.2.2 HTTP Methods

`GET`

7.2.3 Parameters

doc-id	UUID of the document
attachment-id	UUID of the attachment

7.2.4 Formats

JSON

7.2.5 Authentication Required

true

7.2.6 Request Entity

n/a

7.2.7 Responses

200	OK
404	Link not found

7.2.8 Response Entity

```
{
   "values":[
      {
         "key":"foo",
         "user":"foox",
         "value":"bar",
      }
   ],
   "total":1
}
```

7.3 Insert a new pair to an attchment

Post a new pair to the given attachment.

7.3.1 URI

`http://apipath/documents/` doc-id `/attachments/` attachment-id `/values`

7.3.2 HTTP Methods

`POST`

7.3.3 Parameters

doc-id	UUID of the document
attachment-id	UUID of the attachment

7.3.4 Formats

JSON

7.3.5 Authentication Required

true

7.3.6 Request Entity

```
{
   "key":"testkey"
   "value":"testvalue"
}
```

7.3.7 Responses

201	Pair stored
400	Invalid key value pair (i.e. too long value)
404	Link not found
409	Key already exist (use PUT for replace)

7.3.8 Response Entity

n/a

7.4 Return a specific key/value pair of an attachment

Get a specific pair of the given attachment.

7.4.1 URI

`http://apipath/documents/` `doc-id` `/attachments/` `attachment-id` `/values/user/` `user` `/key/` `key`

7.4.2 HTTP Methods

`GET`

7.4.3 Parameters

doc-id	UUID of the document
attachment-id	UUID of the attachment
user	The user ID of the user currently logged in
key	The system-provided key for the value (this is not the plain key of the entry!)

7.4.4 Formats

JSON

7.4.5 Authentication Required

true

7.4.6 Request Entity

n/a

7.4.7 Responses

200	OK
404	Link not found

7.4.8 Response Entity

```
{
   "user":"foox",
   "key":"foo",
   "value": "bar",
   "submitted":"2010-09-10T22:31:08.161Z",
   "valueType":"attachment",
   "doc_id":"06d204be-412d-4be5-bfd9-1ca092f7dd09",
   "attachmentId":"06d204be-412d-4be5-bfd9-1ca092f7dd09"
}
```

7.5 Replace key/value pair of an attachment

Update a pair of the given attachment.

7.5.1 URI

`http://apipath/documents/` `doc-id` `/attachments/` `attachment-id` `/values/user/` `user` `/key/` `key`

7.5.2 HTTP Methods

`PUT`

7.5.3 Parameters

doc-id	UUID of the document
attachment-id	UUID of the attachment
user	The user ID of the user currently logged in
key	The system-provided key for the value (this is not the plain key of the entry!)

7.5.4 Formats

JSON

7.5.5 Authentication Required

true

7.5.6 Request Entity

```
{
   "key":"testkey"
   "value":"testvalue"
}
```

7.5.7 Responses

202	Pair updated
400	Invalid key value pair (i.e. too long value)
403	Not allowed (the owner of the key differs from the updater)
404	Link not found

7.5.8 Response Entity

n/a

7.6 Remove key/value pair from an attachment

Delete a pair of the given attachment.

7.6.1 URI

`http://apipath/documents/`[`doc-id`]`/attachments/`[`attachment-id`]`/values/user/`[`user`]`/key/`[`key`]

7.6.2 HTTP Methods

`DELETE`

7.6.3 Parameters

doc-id	UUID of the document
attachment-id	UUID of the attachment
user	The user ID of the user currently logged in
key	The system-provided key for the value (this is not the plain key of the entry!)

7.6.4 Formats

JSON

7.6.5 Authentication Required

true

7.6.6 Request Entity

n/a

7.6.7 Responses

204	Pair deleted
403	Not allowed (the owner of the key differs from the updater)

7.6.8 Response Entity

n/a

7.7 Return a list of key/values of a link

List all pair of the given link.

7.7.1 URI

`http://apipath/links/link-id/values`

7.7.2 HTTP Methods

`GET`

7.7.3 Parameters

link-id	UUID of the link

7.7.4 Formats

JSON

7.7.5 Authentication Required

true

7.7.6 Request Entity

n/a

7.7.7 Responses

200	OK
404	Link not found

7.7.8 Response Entity

```
{
   "values":[
      {
         "key":"testkey",
         "user":"foox",
         "value":"testvalue"
      },
      {
         "key":"foo",
         "user":"foox",
         "value":"barx",
      }
   ],
   "total":2
}
```

7.8 Insert a new pair to a link

Post a new pair to the given link.

7.8.1 URI

`http://apipath/links/link-id/values`

7.8.2 HTTP Methods

`POST`

7.8.3 Parameters

link-id	UUID of the link

7.8.4 Formats

JSON

7.8.5 Authentication Required

true

7.8.6 Request Entity

```
{
   "key":"testkey"
   "value":"testvalue"
}
```

7.8.7 Responses

201	Pair stored
400	Invalid key value pair (i.e. too long value)
404	Link not found
409	Key already exist (use PUT for replace)

7.8.8 Response Entity

n/a

7.9 Return a specific key/value pair of a link

Get a specifig pair of the given link.

7.9.1 URI

`http://apipath/links/link-id/values/user/user/key/key`

7.9.2 HTTP Methods

`GET`

7.9.3 Parameters

link-id	UUID of the link
user	The user ID of the user currently logged in
key	The system-provided key for the value (this is not the plain key of the entry!)

7.9.4 Formats

JSON

7.9.5 Authentication Required

true

7.9.6 Request Entity

n/a

7.9.7 Responses

200	OK
404	Link not found

7.9.8 Response Entity

```
{
   "user":"foox",
   "key":"foo",
   "value": "bar",
   "submitted":"2010-09-10T22:50:06.283Z",
   "valueType":"link",
   "link_id":"d4777513-7eb6-de3d-6c90-81cd82216f0f"
}
```

7.10 Replace key/value pair of a link

Update a pair of the given link.

7.10.1 URI

`http://apipath/links/link-id/values/user/user/key/key`

7.10.2 HTTP Methods

`PUT`

7.10.3 Parameters

link-id	UUID of the link
user	The user ID of the user currently logged in
key	The system-provided key for the value (this is not the plain key of the entry!)

7.10.4 Formats

JSON

7.10.5 Authentication Required

true

7.10.6 Request Entity

```
{
   "key":"testkey"
   "value":"testvalue"
}
```

7.10.7 Responses

202	Pair updated
400	Invalid key value pair (i.e. too long value)
403	Not allowed (the owner of the key differs from the updater)
404	Link not found

7.10.8 Response Entity

n/a

7.11 Remove key/value pair from a link

Delete a pair of the given link.

7.11.1 URI

`http://apipath/links/link-id/values/user/user/key/key`

7.11.2 HTTP Methods

`DELETE`

7.11.3 Parameters

link-id	UUID of the link
user	The user ID of the user currently logged in
key	The system-provided key for the value (this is not the plain key of the entry!)

7.11.4 Formats

JSON

7.11.5 Authentication Required

true

7.11.6 Request Entity

n/a

7.11.7 Responses

204	Pair deleted
403	Not allowed (the owner of the key differs from the updater)

7.11.8 Response Entity

n/a

8 Links

The following actions provide methods related to interlinking documents. A link is unidirectional connection between two documents. The semantics of a link is a unidirectional trace from the source document to the destination document, thus it represents a 'happens-after' or 'is-the-result-of' semantics.

8.1 Create a new link between to documents

Creates a new link.

8.1.1 URI

`http://apipath/links`

8.1.2 HTTP Methods

`POST`

8.1.3 Parameters

n/a

8.1.4 Formats

JSON

8.1.5 Authentication Required

true

8.1.6 Request Entity

```
1 {
2    "src":"171a90aa-8023-497a-8506-776007e4a8d1",
3    "dest":"1445b334-231b-48e9-a7e0-d06e838f245b",
4    "title":"What happened to the car",
5    "desc":"This image was taken after the car has been put on fire
         "
6 }
```

`src` and `dest` are document IDs of existing documents. `title` is a mandatory heading for the link, `desc` and optional-emtpy description of the link.

8.1.7 Responses

201	Created + Location of generated link
400	Invalid metadata

8.1.8 Response Entity

n/a

8.2 Get a link

Request an existing link by it's id.

8.2.1 URI

`http://apipath/links/` `link-id`

8.2.2 HTTP Methods

`POST`

8.2.3 Parameters

link-id	id of the link

8.2.4 Formats

JSON

8.2.5 Authentication Required

true

8.2.6 Request Entity

n/a

8.2.7 Responses

200	OK
404	Link not found

8.2.8 Response Entity

```
{
   "id":"550e8400-e29b-11d4-a716-446655440000",
   "src":"171a90aa-8023-497a-8506-776007e4a8d1",
   "dest":"1445b334-231b-48e9-a7e0-d06e838f245b",
   "title":"foo bar",
   "desc":"cool stuff",
   "created":"2010-06-26T17:49:26.841Z"
}
```

8.3 List inbound/outbound links of a document

Returns a list of either all inbound or outbound links of a document.

8.3.1 URI

`http://apipath/documents/doc-id/links/(inbound|outbound)`

8.3.2 HTTP Methods

`POST`

8.3.3 Parameters

doc-id	id of the document

8.3.4 Formats

JSON

8.3.5 Authentication Required

true

8.3.6 Request Entity

n/a

8.3.7 Responses

200	OK
404	document not found

8.3.8 Response Entity

```
{
   "links":[
      {
         "src":"171a90aa-8023-497a-8506-776007e4a8d1",
         "dest":"1445b334-231b-48e9-a7e0-d06e838f245b",
         "title":"foo bar",
         "desc":"cool stuff",
         "created":"2010-06-26T17:49:26.841Z",
         "id":"550e8400-e29b-11d4-a716-446655440000"
      }.
      {
         "src":"271a90aa-8023-497a-8506-776007e4a8d1",
         "dest":"3445b334-231b-48e9-a7e0-d06e838f245b",
         "title":"foo bar2",
         "desc":"even cooler stuff",
         "created":"2010-06-26T17:51:26.841Z",
         "id":"660e8400-e29b-11d4-a716-446655440011"
      }
   ]
}
```

9 Document Origin Positions

These methods give access to location-based meta data of documents.

9.1 Get a positions by document

Returns a list of positions of the given document.

9.1.1 URI

`http://apipath/documents/doc-id/positions`

9.1.2 HTTP Methods

`GET`

9.1.3 Parameters

doc-id	UUID of the document

9.1.4 Formats

JSON

9.1.5 Authentication Required

true

9.1.6 Request Entity

n/a

9.1.7 Responses

200	OK
404	Document not found

9.1.8 Response Entity

```
1 {
2    "positions":[
3       {
4          "id":"21f71c7c-8304-4940-a87b-885d5aff1f92",
5          "location":{
```

```
            "position":{
               "type":"Point",
               "coordinates":[
                  12,
                  11
               ]
            },
            "variance":13
         }
      }.
      {
         "id":"550e8400-e29b-11d4-a716-446655440000",
         "location":{
            "position":{
               "type":"Point",
               "coordinates":[
                  13,
                  12
               ]
            },
            "variance":20
         }
      }
   ]
}
```

9.2 Get a position by its id

Returns information about a position data entry by its id.

9.2.1 URI

`http://apipath/documents/` `doc-id` `/positions/` `pos-id` (calls to `http://apipath/positions/` `pos-id` will be 303'ed to the correct URI)

9.2.2 HTTP Methods

`GET`

9.2.3 Parameters

doc-id	UUID of the document
pos-id	UUID of the position

9.2.4 Formats

JSON

9.2.5 Authentication Required

true

9.2.6 Request Entity

n/a

9.2.7 Responses

200	OK
404	Position not found

9.2.8 Response Entity

```
{
   "id":"ffe23dd5-9c5f-4af5-a392-bf597e58b071",
   "docId":"ffe23dd5-9c5f-4af5-a392-bf597e58b071",
   "submitted":"2010-08-05T21:35:28.705Z",
   "user":"630b2cdc55dcbc0169b293cea9ea4234",
   "location":{
      "position":{
         "type":"Point",
         "coordinates":[
            9.9586211594584,
            48.42222094541325
         ]
      },
      "variance":10
   }
}
```

9.3 Get all positions within a bounding box

Returns all matching positions within a given bounding box.

9.3.1 URI

`http://apipath/positions/inside/lat1/lon1/lat2/lon2`

9.3.2 HTTP Methods

GET

9.3.3 Parameters

lat1	Latitude of lower left edge
lon1	longitude of lower left edge
lat2	Latitude of upper right edge
lon2	longitude of upper right edge

9.3.4 Formats

JSON

9.3.5 Authentication Required

true

9.3.6 Request Entity

n/a

9.3.7 Responses

200	OK

9.3.8 Response Entity

```
{
   "positions":[
      {
         "doc_id":"8b5eb8eb-07cc-4701-8023-969346b40ee8",
         "position_id":"8b5eb8eb-07cc-4701-8023-969346b40ee8",
         "location":{
            "position":{
               "type":"Point",
               "coordinates":[
                  10.114013671875,
                  48.15293884277344
               ]
            },
            "variance":2
         }
      },
```

```
    {
       "doc_id":"03f70cdb-4cc3-4622-aaa5-54e8af1b3937",
       "position_id":"03f70cdb-4cc3-4622-aaa5-54e8af1b3937",
       "location":{
          "position":{
             "type":"Point",
             "coordinates":[
                10.113791465759277,
                48.15301513671875
             ]
          },
          "variance":4
       }
    }
 ]
}
```


10 Service Information

This subservice returns runtime informations about the deployed diretto platform.

10.1 Get supported media Formats

Returns data about the media formats supported by this platform.

10.1.1 URI

`http://apipath/service/supportedformats`

10.1.2 HTTP Methods

`GET`

10.1.3 Parameters

n/a

10.1.4 Formats

JSON

10.1.5 Authentication Required

true

10.1.6 Request Entity

n/a

10.1.7 Responses

200 OK	Response entity containing the list

10.1.8 Response Entity

```
{
   "mimetypes":{
      "image/jpeg":{
         "extension":".jpg",
         "maxsize":5242880,
         "type":"image"
      },
      "image/png":{
         "extension":".png",
         "maxsize":5242880,
         "type":"image"
      }
   },
   "types":{
      "image":{
         "traits":[
            "discrete",
            "visual"
         ]
      },
      "text":{
         "traits":[
            "discrete",
            "textual"
         ]
      },
      "video":{
         "traits":[
            "continuous",
            "visual",
            "audible"
```

```
32          ]
33       },
34       "audio":{
35          "traits":[
36             "continuous",
37             "audible"
38          ]
39       }
40    },
41    "traits":[
42       "discrete",
43       "continuous",
44       "textual",
45       "visual",
46       "audible"
47    ]
48 }
```

10.2 Get server info

Returns information about the remote server.

10.2.1 URI

`http://apipath/service/info`

10.2.2 HTTP Methods

`GET`

10.2.3 Parameters

n/a

10.2.4 Formats

JSON

10.2.5 Authentication Required

true

10.2.6 Request Entity

n/a

10.2.7 Responses

200 OK	Response entity containing the info

10.2.8 Response Entity

```
{
   "node":{
      "version":"0.1.97"
   },
   "plattorm":"linux2"
}
```

11 Tags

11.1 Tagable Items

Tags can be added to the following items:

- Attachments
- base URI = `http://apipath/documents/` `doc-id` `/attachments/` `attachment-id`
- Links
- base URI = `http://apipath/links/` `link-id`

11.2 Add a tag to an attachment

Post a new tag to the given attachment.

11.2.1 URI

`http://apipath/documents/` `doc-id` `/attachments/` `attachment-id` `/tags`

11.2.2 HTTP Methods

`POST`

11.2.3 Parameters

doc-id	UUID of the document
attachment-id	UUID of the attachment

11.2.4 Formats

JSON

11.2.5 Authentication Required

true

11.2.6 Request Entity

```
1 {
2    "tag":"mytag"
3 }
```

11.2.7 Responses

201	Tag created
202	Tag already exists, but accepted
400	Invalid tag data (i.e. length)
404	Document/Attachment not found

11.2.8 Response Entity

n/a

11.3 Add a tag to a link

Post a new tag to the given link.

11.3.1 URI

`http://apipath/links/link-id/tags`

11.3.2 HTTP Methods

`POST`

11.3.3 Parameters

link-id	UUID of the link

11.3.4 Formats

JSON

11.3.5 Authentication Required

true

11.3.6 Request Entity

```
{
   "tag":"mytag"
}
```

11.3.7 Responses

201	Tag created
202	Tag already exists, but accepted
400	Invalid tag data (i.e. length)
404	Link not found

11.3.8 Response Entity

n/a

11.4 Get a specific attachment tag by tag ID

Get a tag attached to an attachment by its id.

11.4.1 URI

`http://apipath/documents/` `doc-id` `/attachments/` `attachment-id` `/tags/` `tag-id`

11.4.2 HTTP Methods

`GET`

11.4.3 Parameters

doc-id	UUID of the document
attachment-id	UUID of the attachment
tag-id	ID of the link

11.4.4 Formats

JSON

11.4.5 Authentication Required

true

11.4.6 Request Entity

n/a

11.4.7 Responses

200	OK
404	Link or tag not found

11.4.8 Response Entity

```
1 {
2    "tag_id":"fe7691f6c14d0064ccf9c9bc4a67d240--a-06d204be-412d-4
         be5-bfd9-1ca092f7dd09",
3    "doc_id":"06d204be-412d-4be5-bfd9-1ca092f7dd09",
4    "attachment_id":"06d204be-412d-4be5-bfd9-1ca092f7dd09",
5    "created":"2010-09-10T20:00:37.000Z",
6    "user":"foox",
7    "tag":"testtag"
8 }
```

11.5 Get a specific link tag by tag ID

Get a tag attached to a link by its id.

11.5.1 URI

`http://apipath/links/link-id/tags/tag-id`

11.5.2 HTTP Methods

`GET`

11.5.3 Parameters

link-id	UUID of the link
tag-id	ID of the link

11.5.4 Formats

JSON

11.5.5 Authentication Required

true

11.5.6 Request Entity

n/a

11.5.7 Responses

200	OK
404	Link or tag not found

11.5.8 Response Entity

```
1 {
2    "tag_id":"fe7691f6c14d0064ccf9c9bc4a67d240--l-d4777513-7eb6-
         de3d-6c90-81cd82216f0f",
3    "link_id":"d4777513-7eb6-de3d-6c90-81cd82216f0f",
4    "created":"2010-09-10T20:00:49.507Z",
5    "user":"foox",
6    "tag":"testtag"
7 }
```

11.6 List all tags of an attachment

List all tags of a given attachment.

11.6.1 URI

`http://apipath/documents/` `doc-id` `/attachments/` `attachment-id` `/tags`

11.6.2 HTTP Methods

`GET`

11.6.3 Parameters

doc-id	UUID of the document
attachment-id	UUID of the attachment

11.6.4 Formats

JSON

11.6.5 Authentication Required

true

11.6.6 Request Entity

n/a

11.6.7 Responses

200	List of tags
404	Attachment not found

11.6.8 Response Entity

```
1 {
2    "tags":[
3       {
4          "tag_id":"fe7691f6c14d0064ccf9c9bc4a67d240--l-d4777513-7
               eb6-de3d-6c90-81cd82216f0f",
5          "tag":"testtag"
6       }
7    ],
8    "total":1
9 }
```

11.7 List all tags of a link

List all tags of a given link.

11.7.1 URI

`http://apipath/links/link-id/tags`

11.7.2 HTTP Methods

`GET`

11.7.3 Parameters

link-id	UUID of the link

11.7.4 Formats

JSON

11.7.5 Authentication Required

true

11.7.6 Request Entity

n/a

11.7.7 Responses

20	List of tags
404	Link not found

11.7.8 Response Entity

```
1 {
2    "tags":[
3       {
4          "tag_id":"fe7691f6c14d0064ccf9c9bc4a67d240--l-d4777513-7
               eb6-de3d-6c90-81cd82216f0f",
5          "tag":"testtag"
6       }
7    ],
8    "total":1
9 }
```

12 Document Origin Times

These methods give access to temporal meta data of documents.

12.1 Get a list of times by document

Returns a list of times of the given document.

12.1.1 URI

`http://apipath/documents/doc-id/times`

12.1.2 HTTP Methods

`GET`

12.1.3 Parameters

doc-id	UUID of the document

12.1.4 Formats

JSON

12.1.5 Authentication Required

true

12.1.6 Request Entity

n/a

12.1.7 Responses

200	OK
404	Document not found

12.1.8 Response Entity

```
1 {
2    "times":[
3       {
4          "time_id":"39a2342b-f526-4cc6-abd2-d48ae215cdd6",
5          "submitted":"2010-09-09T19:56:37.992Z",
```

```
6           "user":"630b2cdc55dcbc0169b293cea9ea4234",
7           "date":{
8              "after":"2010-09-09T21:26:37.576Z",
9              "before":"2010-09-09T21:56:37.576Z"
10          }
11       }
12    ]
13 }
```

12.2 Get a time entry by its id

Returns information about a time data entry by its id.

12.2.1 URI

`http://apipath/documents/` `doc-id` `/times/` `time-id` (calls to `http://apipath/times/` `time-id` will be 303'ed to the correct URI)

12.2.2 HTTP Methods

`GET`

12.2.3 Parameters

doc-id	UUID of the document
time-id	UUID of the time

12.2.4 Formats

JSON

12.2.5 Authentication Required

true

12.2.6 Request Entity

n/a

12.2.7 Responses

200	OK
404	Time not found

12.2.8 Response Entity

```
{
   "time_id":"39a2342b-f526-4cc6-abd2-d48ae215cdd6",
   "doc_id":"39a2342b-f526-4cc6-abd2-d48ae215cdd6",
   "submitted":"2010-09-09T19:56:37.992Z",
   "user":"630b2cdc55dcbc0169b293cea9ea4234",
   "date":{
      "after":"2010-09-09T21:26:37.576Z",
      "before":"2010-09-09T21:56:37.576Z"
   }
}
```

12.3 Get all times within a time range

Returns a list of matching times within a given range. Note that exact moments depict the start of a creation, not the end. This is especially important for documents that have a duration. The upper bound is the last possible time the recording has been started, not the time it has ended.

12.3.1 URI

`http://apipath/times/between/` `after` / `before`

12.3.2 HTTP Methods

`GET`

12.3.3 Parameters

after	Lower bound
before	Upper bound

Parameters must be encoded as RFC 3339 Zulu time values.

12.3.4 Formats

JSON

12.3.5 Authentication Required

true

12.3.6 Request Entity

n/a

12.3.7 Responses

200	OK

12.3.8 Response Entity

```
{
   "times":[
      {
         "doc_id":"39a2342b-f526-4cc6-abd2-d48ae215cdd6",
         "time_id":"39a2342b-f526-4cc6-abd2-d48ae215cdd6",
         "date":{
            "after":"2010-09-09T21:26:37.576Z",
            "before":"2010-09-09T21:56:37.576Z"
         }
      }
   ]
}
```

13 User Accounts

The following actions provide methods related to the user accounts and profiles. This version of the API requires every user to be authenticated when committing actions within the system except special actions like account creation. Users are identified by a unique string that can be changed and is bound forever to the user. Currently, the identification string is a hashed version of the mail address. Thus, users are not allowed to change their address yet.

13.1 Create Account

Creates a new user account. Notice that user ids are generated by the server for forward compatibility. In this version, the user id is generated by hashing the mail address via MD5 though.

13.1.1 URI

`http://apipath/users/`

13.1.2 HTTP Methods

`POST`

13.1.3 Parameters

n/a

13.1.4 Formats

JSON

13.1.5 Authentication Required

false

13.1.6 Request Entity

JSON document containing email, sha1(password) and username:

```
1 {
2  "email" : "john@doe.net",
3  "password" : "da39a3ee5e6b4b0d3255bfef95601890afd80709",
4  "username" : "John Doe"
5 }
```

13.1.7 Responses

201 Created + Location Header	User has been created
400 Bad Request	Missing values
403 Forbidden	User account creation has been disabled
409 Conflict	User ID already exists

13.1.8 Response Entity

n/a

13.2 Change Account

Changes an existing user account. This method can only be called by the auth'ed user himself. The changes must not affect the user id, thus we will not allow the user to change his mail address.

13.2.1 URI

`http://apipath/users/userid`

13.2.2 HTTP Methods

`PUT`

13.2.3 Parameters

userid	The internal id of the user

13.2.4 Formats

JSON

13.2.5 Authentication Required

true

13.2.6 Request Entity

JSON document containing email, sha1(password) and username:

```
1 {
2  "email" : "john@doe.net",
3  "password" : "da39a3ee5e6b4b0d3255bfef95601890afd80709",
4  "username" : "John Doe"
5 }
```

13.2.7 Responses

202 Accepted	User profile has been changed
400 Bad Request	Missing values
403 Forbidden	The profile id differs from the id of the user making the call

13.2.8 Response Entity

n/a

13.3 Delete Account

Deletes an account. This method can only be called by the auth'ed user himself.

13.3.1 URI

`http://apipath/users/userid`

13.3.2 HTTP Methods

`DELETE`

13.3.3 Parameters

userid	The internal id of the user

13.3.4 Formats

n/a

13.3.5 Authentication Required

true

13.3.6 Request Entity

n/a

13.3.7 Responses

204 No Content	User profile has been deleted
400 Bad Request	Invalid user id
403 Forbidden	The profile id differs from the id of the user making the call

13.3.8 Response Entity

n/a

13.4 Get Account Data

Provides information about a user except the password.

13.4.1 URI

`http://apipath/users/userid`

13.4.2 HTTP Methods

`GET`

13.4.3 Parameters

userid	The internal id of the user

13.4.4 Formats

JSON

13.4.5 Authentication Required

true

13.4.6 Request Entity

n/a

13.4.7 Responses

200 Accepted	User profile has been changed
404 Not Found	Invalid user id

13.4.8 Response Entity

```
1 {
2  "email" : "john@doe.net",
3  "username" : "John Doe"
4 }
```

14 User List

14.1 List all users

Lists all existing user accounts sorted by user-id. Results are split up into chunks of 100 users. Navigation within the pagination is possible using cursor ids. These ids represent the first entry of a chunk of the previous/following result page. First/last page omits the previous/next key-value pair. A request without cursor will start with the first available value as cursor.

14.1.1 URI

- `http://apipath/users/` =¿ will be 303'ed to `http://apipath/users/cursor/` `firstcursor`
- `http://apipath/users/cursor/` `userid`

14.1.2 HTTP Methods

`GET`

14.1.3 Parameters

n/a

14.1.4 Formats

JSON

14.1.5 Authentication Required

true

14.1.6 Request Entity

n/a

14.1.7 Responses

200 Accepted	List delivered
303 See Other + Location including first cursor	Query without cursor
404 Not found	Cursor not found

14.1.8 Response Entity

```
1  {
2    "users" : [
3      {"id" : "3e64c371ba2b93f1c0fead369fe004ef", "username" : "Max
           Mustermann" },
4      //98 other entries here...
5      {"id" : "57ebb75106e99f6e7299fefe37de83b7", "username" : "John
           Doe" }
6    ],
7    "next" : "acbd18db4cc2f85cedef654fccc4a4d8",
8    "previous" : "37b51d194a7513e45b56f6524f2d51f2",
9    "total" : 1234
10 }
```

15 Votes

15.1 Votable Items

Voting can be applied to various items. The following list shows the base URI of a votable entry, the methods below how to interact with them.

- Attachments
- base URI = `http://apipath/documents/` `doc-id` `/attachments/` `attachment-id`
- Links
- base URI = `http://apipath/links/` `link-id`
- Comments (for links)
- base URI = `http://apipath/links/` `link-id` `/comments/` `comment-id`
- Comments (for attachments)
- base URI = `http://apipath/documents/` `doc-id` `/attachments/` `attachment-id` `/comments/` `comment`
- Tags (for links)
- base URI = `http://apipath/links/` `link-id` `/tags/` `tag-id`
- Tags (for attachments)
- base URI = `http://apipath/documents/` `doc-id` `/position/` `attachment-id` `/tags/` `tag-id`
- Position
- base URI = `http://apipath/documents/` `doc-id` `/position/` `position-id`
- Times
- base URI = `http://apipath/documents/` `doc-id` `/times/` `time-id`

15.2 Cast a vote

15.2.1 URI

`http://apipath/entryPath/vote/` `vote`

15.2.2 HTTP Methods

`GET`

15.2.3 Parameters

entry path	The real path to the entry (see votable entries)
vote	”up” or ”down”

15.2.4 Formats

JSON

15.2.5 Authentication Required

true

15.2.6 Request Entity

n/a

15.2.7 Responses

201	First vote casted
202	Vote changed

15.2.8 Response Entity

n/a

15.3 Undo a vote

15.3.1 URI

`http://apipath/entryPath/vote`

15.3.2 HTTP Methods

`DELETE`

15.3.3 Parameters

entry path	The real path to the entry (see votable entries)

15.3.4 Formats

JSON

15.3.5 Authentication Required

true

15.3.6 Request Entity

n/a

15.3.7 Responses

204	Vote removed
404	No vote found

15.3.8 Response Entity

n/a

15.4 Get current voting results

15.4.1 URI

`http://apipath/entryPath/votes`

15.4.2 HTTP Methods

`GET`

15.4.3 Parameters

entry path	The real path to the entry (see votable entries)

15.4.4 Formats

JSON

15.4.5 Authentication Required

true

15.4.6 Request Entity

n/a

15.4.7 Responses

200	Vote list in body
404	No vote found

15.4.8 Response Entity

```
{
   "votes":{
      "user":-1,
      "up":21,
      "down":3
   }
}
```

- `user` represents the vote of the user currently logged in. -1 represents a down vote, 1 an up vote and 0 shows that the user has no casted vote yet.
- `up` and `down` sum up all votes in favor our against the entry in total. If there is a vote of the current user, it is also included in the sum.

16 RESTful API for Media Node

This documentation assumes that the root of the media storage is `http://mediastorage/`.

16.1 Retrieve Document Size/Type/ETag

Returns the header information of a GET omitting the body. Helpful for premature access on data like content size, content type or ETag of the item

16.1.1 URI

`http://mediastorage/` `DOC-ID` / `ATTACH-ID`

16.1.2 HTTP Methods

`HEAD`

16.1.3 Parameters

DOC-ID	Identifier for document directory
ATTACH-ID	Identifier for actual item

16.1.4 Formats

n/a

16.1.5 Authentication Required

false

16.1.6 Request Entity

n/a

16.1.7 Responses

200	Successful request
403	The requested path contains forbidden elements
404	The item does not exist
500	The server has encountered an problem while reading/streaming the item

16.1.8 Response Entity

n/a

16.2 Retrieve Item

Downloads a media item from the media storage.

16.2.1 URI

`http://mediastorage/`DOC-ID`/`ATTACH-ID

16.2.2 HTTP Methods

`GET`

16.2.3 Parameters

DOC-ID	Identifier for document directory
ATTACH-ID	Identifier for actual item

16.2.4 Formats

n/a

16.2.5 Authentication Required

false

16.2.6 Request Entity

n/a

16.2.7 Responses

200	Item in body
304	Item is already cached by client
403	The requested path contains forbidden elements
404	The item does not exist
500	The server has encountered an problem while reading/streaming the item

16.2.8 Response Entity

The requested item.

16.3 Upload Item

Uploads a media item to the media storage. "'Please note:"' This method is only valid if there is a corret key specified for the operation. In order to obtain such a key, the client must upload meta data to the core API endpoint prior to this. See also [wiki:Components/API/v1/Workflows/ Workflows].

16.3.1 URI

`http://mediastorage/`DOC-ID`/`ATTACH-ID`?key=`ACCESS-KEY

16.3.2 HTTP Methods

`PUT`

16.3.3 Parameters

DOC-ID	Identifier for document directory
ATTACH-ID	Identifier for actual item
ACCESS-KEY	Key that signs this request

16.3.4 Formats

json

16.3.5 Authentication Required

true

16.3.6 Request Entity

The item to be uploaded.

16.3.7 Responses

201	Item has been created
400	The content was rejected, i.e. due to invalid size
401	Missing or invalid credentials
403	The requested path contains forbidden elements, or the access key is missing/invalid
409	The item already exists
500	The server has encountered an internal problem during upload

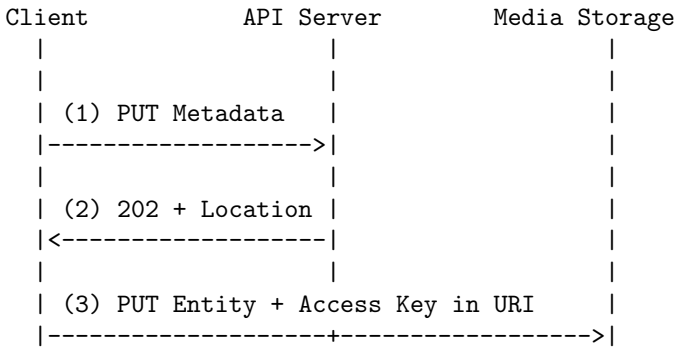
16.3.8 Response Entity

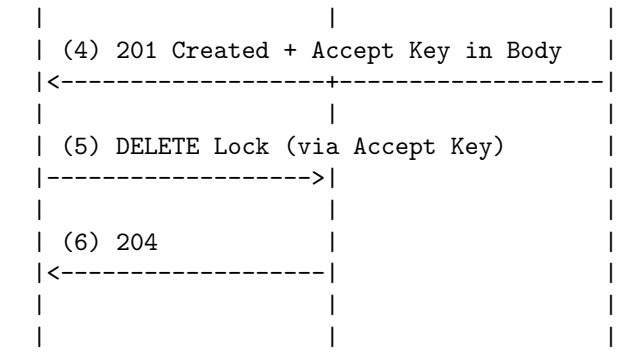
```
1 {
2  "key" : "9365929d8479d2ca39826878984ff5f2b70b5d65"
3 }
```

17 Workflows

Most of the actions that can be conducted via the diretto API are simple and quite self explanatory. However, some actions require multiple interactions. These actions are described here.

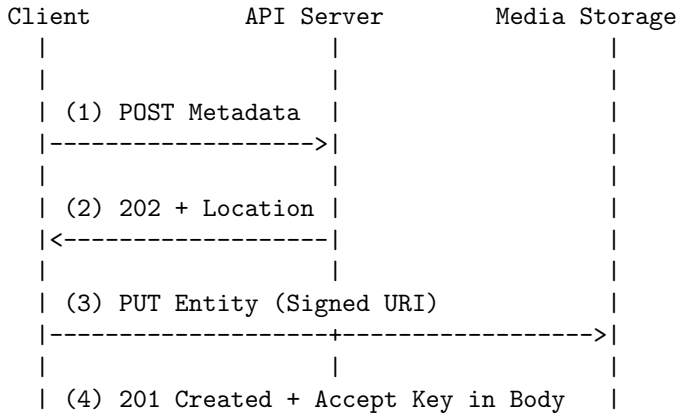
17.1 Upload new Document

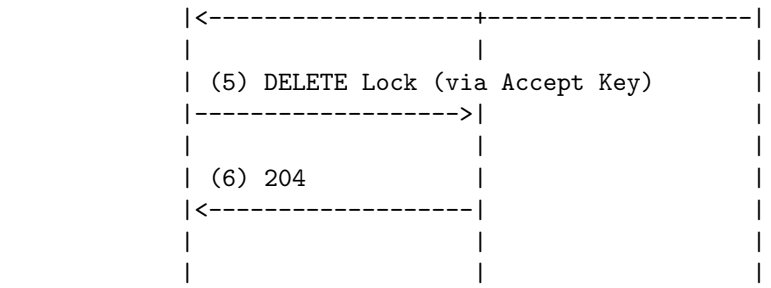




- (1) Upload the metadata information to the API server via `PUT .../documents/doc-id` + meta data in JSON format as entity.
- (2) If the server accepts media type, size etc. and the item does not yet exist, it will return a `202` and a JSON message containing a location, where to upload the item to. This URI should be signed in order to prevent abusive uploads.
- (3) The client now can upload the item to the media storage by `PUT`ting it to the URI provided by the APIs server.
- (4) The server returns 201 if the item has been stored successfully. The entity body contains a JSON message including a signed accept key provided by the storage.
- (5) The client proves the successful upload by deleting the lock using the received accept key: `DELETE .../documents/doc-id/attachments/attach-id/lock/accept-key`
- (6) The server unlocks the item globally.

17.2 Upload new Attachment





- (1) Upload the metadata information to the API server via `POST .../documents/doc-id/attachments/` + meta data in JSON format as entity.
- (2) If the server accepts media type, size etc. and the item does not yet exist, it will return a `202` and a JSON message containing a location, where to upload the item to. It will also return the assigned UUID for the attachment. This URI should be signed in order to prevent abusive uploads.
- (3) The client now can upload the item to the media storage by `PUT`ting it to the URI provided by the APIs server.
- (4) The server returns 201 if the item has been stored successfully. The entity body contains a JSON message including a signed accept key provided by the storage.
- (5) The client proves the successful upload by deleting the lock using the received accept key: `DELETE .../documents/doc-id/attachments/attach-id/lock/accept-key`
- (6) The server unlocks the item globally.