diretto Storage API

Benjamin Erb

Version v2 - Draft

Description

The Storage API is responsible for the reception, storage and delivery of submitted attachments of documents. The Storage API is designed to support HTTP mechanisms like caching and conditional GETs for improved performance.

Contents

Introduction

Supported authentication schemes:

- Basic Authentication

Base URI: http://mediaserver/

1 Attachment

An attachment is a media file that belongs to an abstract document. Each document owns one or more attachments. The first attachment should be considered as the ”original” item, while all additional attachments represent derivated or customized versions.

`http://mediaserver/`$\boxed{\texttt{document-id}}$`/`$\boxed{\texttt{attachment-id}}$`.`$\boxed{\texttt{attachment-ext}}$

document-id	document ID
attachment-id	attachment ID
attachment-ext	attachment extension

1.1 Download an attachment

This operation allows to download an existing attachment. Conditional GETs must be supported by the storage server. Clients should support caching and dispatch conditional requests whenever possible.

URI

`http://mediaserver/`$\boxed{\texttt{document-id}}$`/`$\boxed{\texttt{attachment-id}}$`.`$\boxed{\texttt{attachment-ext}}$

Parameters

HTTP Method

`GET`

Formats

- `varying`

Authentication

not required

Request Entities

n/a

Responses

`200 OK` Attachment in response entity

`304 Not Modified` Cache hit (conditional request)

`403 Forbidden` The requested resource URI is not allowed

`404 Not Found` Attachment and/or document not found

`500 Internal Server Error` Internal error while streaming the attachment

Response Entities

n/a

1.2 Upload a new attachment

New attachments must be uploaded to the storage server as part of a three-part workflow. First, meta data must be sent to the API server. When the API server grants the creation of a new attachment, it also provides an access token that is needed for the actual upload against the storage server. Without a valid access token, no upload can be performed. On succcessful upload, the storage server also creates a token that must be sent to the API server to approve the upload process. This operation does not support chunked uploads and the following request headers are mandatory:

`Content-Type` (mime type), `Content-Length` (file size) and authentication. The actual media file must be provided as it is in the request entity. Please note that creating an initial attachment includes creating a document itself. Thus, the first API call slightly differs, because a `PUT` against the targeted document location is required instead of a `POST` . The complete flow is listed here:

```
  Client         API Server      Media Storage


    |                  |                 |
    |                  |                 |
    | (1) POST Metadata |                |
    |------------------->|               |
    |                  |                 |
    | (2) 202 + Location |               |
    |<-------------------|               |
    |                  |                 |
    | (3) PUT Entity (Signed URI)        |
    |-------------------+------------------>|
    |                  |                 |
    | (4) 201 Created + Accept Key in Body |
    |<------------------+-------------------|
    |                  |                 |
    | (5) DELETE Lock (via Accept Key)   |
    |------------------->|               |
    |                  |                 |
    | (6) 204          |                 |
    |<-------------------|               |
    |                  |                 |
    |                  |                 |
```

URI

`http://mediaserver/` `document-id` `/` `attachment-id` `.` `attachment-ext` `?token=` `token`

Parameters

token	access token

HTTP Method

`PUT`

Formats

- `varying`
- `application/json`

Authentication

required

Request Entities

n/a

Responses

`201 Created` The attachment has been created

`400 Bad Request` The upload has been rejected (i.e. due to invalid file size)

`401 Unauthorized` Wrong or invalid credentials

`403 Forbidden` The targeted path or the provided access token is invalid

`409 Conflict` The media file already exists

`411 Length Required` Upload failed due to missing Content-Length information

`500 Internal Server Error` Internal server error while handling the upload

Response Entities

”successToken” : ”9365929d8479d2ca39826878984ff5f2b70b5d65”

```
"successToken" :  "9365929d8479d2ca39826878984ff5f2b70b5d65"
```

In case of a successful upload (201), a response key is provided that must be send to the API server in order to prove the upload. ”error” : ”The attachment size exceeds the server limits for uploads”

```
"error" :  "The attachment size exceeds the server limits for
        uploads"
```

Otherwise, an error message is available as entity.

1.3 Retrieve attachment’s meta data

A HEAD request against an attachment allows to retrieve several meta data. Servers must provide the following HTTP headers: `Content-Type` (mime type), `Content-Length` (file size) and `ETag` (caching identifier). This operation should also be used for checking the existence of an attachment.

URI

`http://mediaserver/`$\boxed{\texttt{document-id}}$`/`$\boxed{\texttt{attachment-id}}$`.`$\boxed{\texttt{attachment-ext}}$

Parameters

HTTP Method

`HEAD`

Formats

- `n/a`

Authentication

not required

Request Entities

n/a

Responses

`200 OK` Attachment exists, contents omitted

`304 Not Modified` Cache hit (conditional request)

`403 Forbidden` The requested resource URI is not allowed

`404 Not Found` Attachment and/or document not found

`500 Internal Server Error` Internal error while streaming the attachment

Response Entities

n/a

2 Index

The index resource provides static information about the deployed service.
`http://mediaserver/`

2.1 Get static information

This operation allows to receive static information about the deployment.
`HEAD` requests against this resource may be useful to check the service's availability.

URI

```
http://mediaserver/
```

Parameters

HTTP Method

`GET`

Formats

- `application/json`

Authentication

not required

Request Entities

n/a

Responses

`200 OK` Service information in response entity

Response Entities

"api":
"name":"org.diretto.api.storage",
"version":"v2"
,
"service":
"name":"diretto Media Node",
"version":"0.2.0"

```
"api":"name":"org.diretto.api.storage","version":"v2","service":"name":"diretto
         Media Node","version":"0.2.0"
```