Fabasoft

White Paper

Fabasoft OneGov Public API

2026 August Release

Contents

1 Introduction

This guide documents the **Fabasoft OneGov Public API**, a REST API for managing users, teams and fileplan entries in a Fabasoft OneGov organisation. It is written for a Python or Java developer who needs to call this API from a third-party application against an already provisioned OpenAPI service — it covers authentication, every available operation, and how to interpret every response, without requiring you to read the underlying OpenAPI specification yourself.

2 Endpoint Overview

The API exposes the following operations, grouped by path segment:

Method	Path	Summary
POST	`/users`	Create a new user
GET	`/users/{userId}`	Find a user by their email address
DELETE	`/users/{userId}`	Deactivate an existing user
POST	`/teams`	Create a new team with assigned members
POST	`/fileplanentries`	Create a new fileplan entry
GET	`/fileplanentries/{fpeId}`	Get a fileplan entry by its identifier
POST	`/fileplanentries/{fpeId}`	Update attributes of an existing fileplan entry
DELETE	`/fileplanentries/{fpeId}`	Delete a fileplan entry or fileplan entry room by its identifier

3 Service Access

Every Fabasoft OpenAPI service is addressed through a URL of the form `https://<system>/openapi/<serviceCoo>/<path-from-the-specification>`.

`<serviceCoo>` is the COO address of the OpenAPI service itself. It is assigned by Fabasoft when the service is created and stays fixed for the lifetime of that service — the same `serviceCoo` value prefixes every path in the specification, for every operation. It is not chosen or computed by the API consumer. For this public API, the concrete `serviceCoo` value is **`COO.111.100.1.2285060`** — every example in this document uses this literal value as the `SERVICE_COO` constant, rather than a generic placeholder.

Authentication is HTTP Basic Auth using an **"Access for Applications" password**, generated per OpenAPI service (account menu → Advanced Settings → Access for Applications) and valid only for that one service.

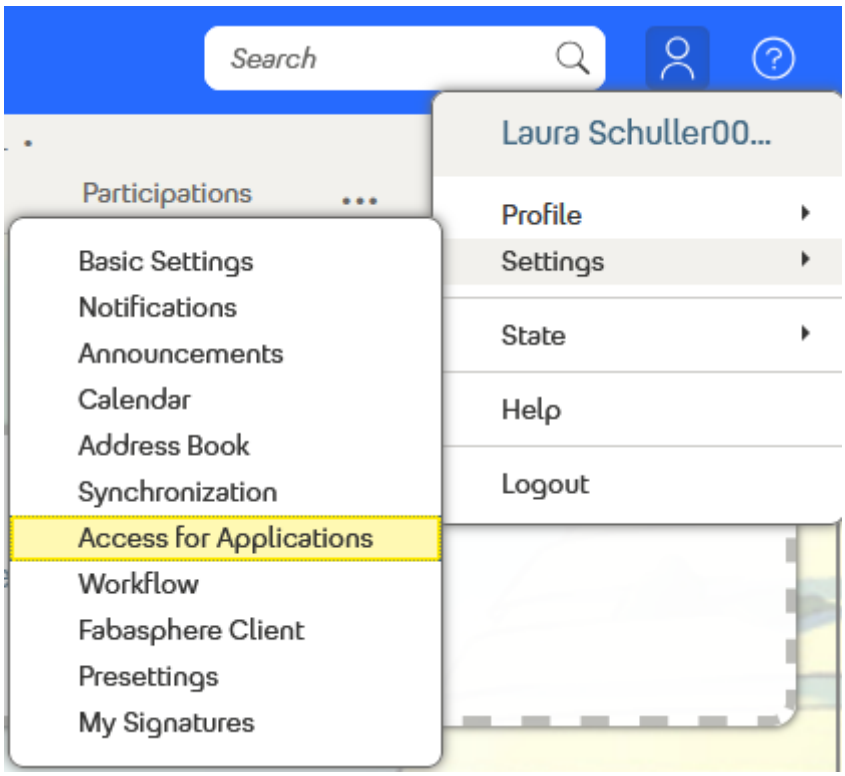

Figure: Locating "Access for Applications" in the account menu.

Shared constants used in every example

All Python and Java examples in this document share the following constants:

Python — shared constants

```
BASE_URL = "https://<system>/openapi"
SERVICE_COO = "COO.111.100.1.2285060"
AUTH = ("<application-user>", "<access-for-applications-password>")
```

Java — shared constants

```
static final String BASE_URL = "https://<system>/openapi";
static final String SERVICE_COO = "COO.111.100.1.2285060";
static final String AUTH_USER = "<application-user>";
static final String AUTH_PASSWORD = "<access-for-applications-password>";
```

4 General Rules

Supported HTTP methods

`PATCH` is currently not supported by the API, instead returning `405 METHOD NOT ALLOWED` when used. Please use `POST` for these endpoints for the time being, as described in the specification.

5 API Reference

5.1 Users

5.1.1 POST /users

Creates one new user. The user state can be deactivated.

Request Body

The request body has the following fields:

Field	Type	Required	Description
`email`	string	Yes	Email address of the user.
`externalkey`	string	No	External identifier for the user in your own system, e.g. an ID from an external HR/user-management system.
`firstname`	string	Yes	First name.
`lastname`	string	Yes	Last name.
`solutions`	array of solution objects	Yes	List of solutions/licenses to assign to the user — see the table below.

Each entry of `solutions[]` has the following fields:

Field	Type	Required	Description
`solution`	string	Yes	Name of the solution, e.g. `Fabasoft OneGov`.
`license`	string	Yes	Type of license to assign, e.g. `Full Access`.

Responses

201 — Successful operation

Field	Type	Required	Description
`id`	string	No[1]	COO address of the newly created user.
`externalkey`	string	No[1]	External key of the user, if one was supplied.
`email`	string	No[1]	Email address of the user.
`firstname`	string	No[1]	First name of the user.
`lastname`	string	No[1]	Last name of the user.

[1] Not marked as required in the schema, but the server always populates this field.

400 — User error

Field	Type	Required	Description
`type`	string	No[1]	Identifier of the error, e.g. `invalid-request-body`.
`title`	string	No[1]	Short summary of the error.
`detail`	string	No[1]	Detailed explanation of the error.
`status`	integer	No[1]	HTTP status code, repeated in the body.

[1] Not marked as required in the schema, but the server always populates this field.

404 — No User Found

Same `GWSError` structure as described above for this operation — see that table for the field explanations.

500 — Server error

Same `GWSError` structure as described above for this operation — see that table for the field explanations.

Python Example

Python example

```
import requests

url = f"{BASE_URL}/{SERVICE_COO}/users"
payload = {
    "email": "hugo.boss@example.com",
    "firstname": "Hugo",
    "lastname": "Boss",
    "externalkey": "extkey-123-49292",
    "solutions": [
        {"solution": "Fabasoft OneGov", "license": "Full Access"}
    ],
}

response = requests.post(url, json=payload, auth=AUTH)
print(response.status_code)
print(response.json())
```

Java Example

Java example

```
import java.net.URI;
import java.net.http.*;
import java.util.Base64;

String auth = Base64.getEncoder().encodeToString((AUTH_USER + ":" +
AUTH_PASSWORD).getBytes());
String body = """
    {
      "email": "hugo.boss@example.com",
      "firstname": "Hugo",
      "lastname": "Boss",
      "externalkey": "extkey-123-49292",
      "solutions": [
        {"solution": "Fabasoft OneGov", "license": "Full Access"}
      ]
    }
    """;

HttpClient client = HttpClient.newHttpClient();
HttpRequest request = HttpRequest.newBuilder()
    .uri(URI.create(BASE_URL + "/" + SERVICE_COO + "/users"))
    .header("Authorization", "Basic " + auth)
    .header("Content-Type", "application/json")
    .POST(HttpRequest.BodyPublishers.ofString(body))
    .build();
```

```
HttpResponse<String> response = client.send(request,
HttpResponse.BodyHandlers.ofString());
System.out.println(response.statusCode());
System.out.println(response.body());
```

5.1.2 DELETE /users/{userId}

Deactivate an existing user.

Parameters

Name	In	Required	Type	Description
`userId`	path	Yes	string	Email, externalkey or COO address of the user, e.g. `hugo.boss@example.com`.

Responses

204 — Successful operation

User was deactivated and licenses are available again. *No response body.*

400 — The user is already deactivated

Field	Type	Required	Description
`type`	string	No[1]	Identifier of the error, e.g. `invalid-request-body`.
`title`	string	No[1]	Short summary of the error.
`detail`	string	No[1]	Detailed explanation of the error.
`status`	integer	No[1]	HTTP status code, repeated in the body.

[1] Not marked as required in the schema, but the server always populates this field.

404 — No user was found

Same `GWSError` structure as described above for this operation — see that table for the field explanations.

500 — Internal server error

Same `GWSError` structure as described above for this operation — see that table for the field explanations.

Python Example

Python example

```
import requests

user_id = "hugo.boss@example.com"
url = f"{BASE_URL}/{SERVICE_COO}/users/{user_id}"

response = requests.delete(url, auth=AUTH)
print(response.status_code)
```

Java Example

Java example

```
import java.net.URI;
import java.net.http.*;
import java.util.Base64;

String auth = Base64.getEncoder().encodeToString((AUTH_USER + ":" +
AUTH_PASSWORD).getBytes());
String userId = "hugo.boss@example.com";

HttpClient client = HttpClient.newHttpClient();
HttpRequest request = HttpRequest.newBuilder()
    .uri(URI.create(BASE_URL + "/" + SERVICE_COO + "/users/" + userId))
    .header("Authorization", "Basic " + auth)
    .DELETE()
    .build();

HttpResponse<String> response = client.send(request,
HttpResponse.BodyHandlers.ofString());
System.out.println(response.statusCode());
```

5.1.3 GET /users/{userId}

Returns a single user, looked up by their email address.

Parameters

Name	In	Required	Type	Description
`userId`	path	Yes	string	Email of the user, e.g. `hugo.boss@example.com`.

Responses

200 — Successful operation

Field	Type	Required	Description
`id`	string	No[1]	COO address of the user.
`externalkey`	string	No[1]	External key of the user.
`email`	string	No[1]	Email address of the user.
`firstname`	string	No[1]	First name of the user.
`lastname`	string	No[1]	Last name of the user.

[1] Not marked as required in the schema, but the server always populates this field.

404 — No user was found

Field	Type	Required	Description
`type`	string	No[1]	Identifier of the error, e.g. `invalid-request-body`.
`title`	string	No[1]	Short summary of the error.
`detail`	string	No[1]	Detailed explanation of the error.
`status`	integer	No[1]	HTTP status code, repeated in the body.

[1] Not marked as required in the schema, but the server always populates this field.

Python Example

Python example

```
import requests

user_id = "hugo.boss@example.com"
url = f"{BASE_URL}/{SERVICE_COO}/users/{user_id}"

response = requests.get(url, auth=AUTH)
print(response.status_code)
```

```
print(response.json())
```

Java Example

Java example

```
import java.net.URI;
import java.net.http.*;
import java.util.Base64;

String auth = Base64.getEncoder().encodeToString((AUTH_USER + ":" +
AUTH_PASSWORD).getBytes());
String userId = "hugo.boss@example.com";

HttpClient client = HttpClient.newHttpClient();
HttpRequest request = HttpRequest.newBuilder()
    .uri(URI.create(BASE_URL + "/" + SERVICE_COO + "/users/" + userId))
    .header("Authorization", "Basic " + auth)
    .GET()
    .build();

HttpResponse<String> response = client.send(request,
HttpResponse.BodyHandlers.ofString());
System.out.println(response.statusCode());
System.out.println(response.body());
```

5.2 Teams

5.2.1 POST /teams

Creates a new team.

Request Body

The request body has the following fields:

Field	Type	Required	Description
`name`	string	Yes	Name of the team.
`externalkey`	string	Yes	External key of the team. Must be unique in the organisation.
`members`	array of string	No	List of users to add as team members. Each entry can be the COO address (`id`), email address, or externalkey of an existing user.

Responses

201 — Successful operation

Field	Type	Required	Description
`id`	string	No[1]	COO address of the newly created team.
`externalkey`	string	No[1]	External key of the team.
`name`	string	No[1]	Name of the team.
`members`	array of user objects	No[1]	The team's members. Same structure as the response described under **POST /users → Responses → 201**.

[1] Not marked as required in the schema, but the server always populates this field.

400 — User error

Field	Type	Required	Description
`type`	string	No[1]	Identifier of the error type, e.g. `invalid-request-body`.
`title`	string	No[1]	Short summary of the error.
`status`	integer	No[1]	HTTP status code, repeated in the body.
`errors`	array of error segments	No[1]	One entry per individual validation problem — see the `errors[]` table below.

[1] Not marked as required in the schema, but the server always populates this field.

Each entry of `errors[]` has the following fields:

Field	Type	Required	Description
`detail`	string	No[1]	Human-readable explanation of this specific validation error.
`pointer`	string	No[1]	JSON Pointer (e.g. `#/name`) identifying which request field the error refers to.

[1] Not marked as required in the schema, but the server always populates this field.

500 — Server error

Field	Type	Required	Description
`type`	string	No[1]	Identifier of the error, e.g. `invalid-request-body`.
`title`	string	No[1]	Short summary of the error.
`detail`	string	No[1]	Detailed explanation of the error.
`status`	integer	No[1]	HTTP status code, repeated in the body.

[1] Not marked as required in the schema, but the server always populates this field.

Python Example

Python example

```
import requests

url = f"{BASE_URL}/{SERVICE_COO}/teams"
payload = {
    "name": "IT Department",
    "externalkey": "it-dept-01",
    "members": [
        "hugo.boss@example.com",
        "extkey-123-49292",
    ],
}

response = requests.post(url, json=payload, auth=AUTH)
print(response.status_code)
print(response.json())
```

Java Example

Java example

```
import java.net.URI;
import java.net.http.*;
import java.util.Base64;

String auth = Base64.getEncoder().encodeToString((AUTH_USER + ":" +
AUTH_PASSWORD).getBytes());
String body = """
```

```
    {
      "name": "IT Department",
      "externalkey": "it-dept-01",
      "members": ["hugo.boss@example.com", "extkey-123-49292"]
    }
    """;

HttpClient client = HttpClient.newHttpClient();
HttpRequest request = HttpRequest.newBuilder()
    .uri(URI.create(BASE_URL + "/" + SERVICE_COO + "/teams"))
    .header("Authorization", "Basic " + auth)
    .header("Content-Type", "application/json")
    .POST(HttpRequest.BodyPublishers.ofString(body))
    .build();

HttpResponse<String> response = client.send(request,
HttpResponse.BodyHandlers.ofString());
System.out.println(response.statusCode());
System.out.println(response.body());
```

5.3 Fileplan Entries

5.3.1 POST /fileplanentries

Creates a fileplan entry.

Request Body

The request body has the following fields:

Field	Type	Required	Description
`title`	string	Yes	German title of the fileplan entry.
`description`	string	No	German description of the fileplan entry.
`primaryrelated`	string	Yes	COO address or external key (`objexternalkey`) of an existing fileplan, fileplan entry, or fileplan entry room under which this new entry will be created.
`externalkey`	string	No	External key of the fileplan entry. Must be unique in the organisation.
`classification`	string	No	Classification of the fileplan entry. Allowed values (case-insensitive): `CONFIDENTIAL`, `CLASSIFIED`, `UNPROTECTED`.

`privacyprotection`	boolean	No	Whether the fileplan entry is privacy protected.
`disclosurestatus`	string	No	Status of the disclosure. Allowed values (case-insensitive): `PUBLIC`, `LIMITEDPUBLIC`, `PRIVATE`, `NOTASSESSED`, `NOFOIA`.
`disclosurestatusstatement`	string	No	Free-text statement explaining the chosen disclosure status.
`retentionperiod`	integer	No	Retention period in years.
`retentionperiodcomment`	string	No	Comment for the retention period.
`archivalvalue`	string	No	Decision whether the fileplan entry is predestined to be archived. Allowed values (case-insensitive): `NOTASSESSED`, `PROMPT`, `ARCHIVALWORTHY`, `NOTARCHIVALWORTHY`, `SAMPLING`.
`archivalvaluecomment`	string	No	Comment for the archival value.
`regularsafeguardperiod`	integer	No	Safeguard period in years — the time period for which the fileplan entry is retained in the archive.
`objsecsecurity`	array of string	No	List of users granted admin/security rights on the fileplan entry. Each entry can be the COO address (`id`), email address, or externalkey of a user.
`objsecchange`	array of string	No	List of users granted change access. Same identifier rules as `objsecsecurity`.
`objsecread`	array of string	No	List of users granted read access. Same identifier rules as `objsecsecurity`.

Responses

201 — Successful operation

Field	Type	Required	Description
`id`	string	No[1]	COO address of the newly created fileplan entry.

`externalkey`	string	No[2]	External key of the newly created fileplan entry, if one was supplied.
`name`	string	No[1]	German name of the fileplan entry.

[1] Not marked as required in the schema, but the server always populates this field. [2] Not marked as required in the schema; the server always includes this field, but its value can be `null`.

400 — User error

Field	Type	Required	Description
`type`	string	No[1]	Identifier of the error type, e.g. `invalid-request-body`.
`title`	string	No[1]	Short summary of the error.
`status`	integer	No[1]	HTTP status code, repeated in the body.
`errors`	array of error segments	No[1]	One entry per individual validation problem — see the `errors[]` table below.

[1] Not marked as required in the schema, but the server always populates this field.

Each entry of `errors[]` has the following fields:

Field	Type	Required	Description
`detail`	string	No[1]	Human-readable explanation of this specific validation error.
`pointer`	string	No[1]	JSON Pointer (e.g. `#/name`) identifying which request field the error refers to.

[1] Not marked as required in the schema, but the server always populates this field.

500 — Server error

Field	Type	Required	Description
`type`	string	No[1]	Identifier of the error, e.g. `invalid-request-body`.
`title`	string	No[1]	Short summary of the error.

`detail`	string	No[1]	Detailed explanation of the error.
`status`	integer	No[1]	HTTP status code, repeated in the body.

[1] Not marked as required in the schema, but the server always populates this field.

Python Example

Python example

```
import requests

url = f"{BASE_URL}/{SERVICE_COO}/fileplanentries"
payload = {
    "title": "Gemeinderat",
    "primaryrelated": "COO.1.30000.1.3662",
    "description": "Sitzungen und Beschluesse des Gemeinderats",
    "classification": "CONFIDENTIAL",
    "disclosurestatus": "PRIVATE",
}

response = requests.post(url, json=payload, auth=AUTH)
print(response.status_code)
print(response.json())
```

Java Example

Java example

```
import java.net.URI;
import java.net.http.*;
import java.util.Base64;

String auth = Base64.getEncoder().encodeToString((AUTH_USER + ":" +
AUTH_PASSWORD).getBytes());
String body = """
    {
      "title": "Gemeinderat",
      "primaryrelated": "COO.1.30000.1.3662",
      "description": "Sitzungen und Beschluesse des Gemeinderats",
      "classification": "CONFIDENTIAL",
      "disclosurestatus": "PRIVATE"
    }
    """;
```

```
HttpClient client = HttpClient.newHttpClient();
HttpRequest request = HttpRequest.newBuilder()
    .uri(URI.create(BASE_URL + "/" + SERVICE_COO + "/fileplanentries"))
    .header("Authorization", "Basic " + auth)
    .header("Content-Type", "application/json")
    .POST(HttpRequest.BodyPublishers.ofString(body))
    .build();

HttpResponse<String> response = client.send(request,
HttpResponse.BodyHandlers.ofString());
System.out.println(response.statusCode());
System.out.println(response.body());
```

5.3.2 DELETE /fileplanentries/{fpeId}

Deletes a fileplan entry or fileplan entry room. Only empty fileplan entries and fileplan entry rooms — i.e. ones without children — can be deleted.

Parameters

Name	In	Required	Type	Description
`fpeId` (COO address)	path	Yes	string	COO address of a fileplan entry (room), e.g. `COO.1.30000.1.1234`. COO addresses are always unique on the system.
`fpeId` (externalkey)	path	Yes	string	Externalkey of a fileplan entry (room), e.g. `cf14d3a0-72ba-45f4-8814-3035d5608708`. Externalkeys are not guaranteed to be unique on the system.

Responses

204 — Successful operation

No response body.

400 — The fileplan entry (room) has child objects and cannot be deleted

Field	Type	Required	Description
`type`	string	No[1]	Identifier of the error, e.g. `invalid-request-body`.
`title`	string	No[1]	Short summary of the error.
`detail`	string	No[1]	Detailed explanation of the error.

`status`	integer	No[1]	HTTP status code, repeated in the body.

[1] Not marked as required in the schema, but the server always populates this field.

403 — Forbidden operation

Same `GWSError` structure as described above for this operation — see that table for the field explanations.

404 — No fileplan entry (room) was found

Same `GWSError` structure as described above for this operation — see that table for the field explanations.

Python Example

Python example

```
import requests

fpe_id = "COO.1.30000.1.1234"  # COO address or externalkey of a fileplan entry
(room)
url = f"{BASE_URL}/{SERVICE_COO}/fileplanentries/{fpe_id}"

response = requests.delete(url, auth=AUTH)
print(response.status_code)
```

Java Example

Java example

```
import java.net.URI;
import java.net.http.*;
import java.util.Base64;

String auth = Base64.getEncoder().encodeToString((AUTH_USER + ":" +
AUTH_PASSWORD).getBytes());
String fpeId = "COO.1.30000.1.1234"; // COO address or externalkey of a fileplan
entry (room)

HttpClient client = HttpClient.newHttpClient();
HttpRequest request = HttpRequest.newBuilder()
    .uri(URI.create(BASE_URL + "/" + SERVICE_COO + "/fileplanentries/" + fpeId))
    .header("Authorization", "Basic " + auth)
    .DELETE()
    .build();
```

```
HttpResponse<String> response = client.send(request,
HttpResponse.BodyHandlers.ofString());
System.out.println(response.statusCode());
```

5.3.3 GET /fileplanentries/{fpeId}

Returns a single fileplan entry.

Parameters

Name	In	Required	Type	Description
`fpeId` (COO address)	path	Yes	string	COO address of a fileplan entry, e.g. `COO.1.30000.1.1234`. COO addresses are always unique on the system.
`fpeId` (externalkey)	path	Yes	string	Externalkey of a fileplan entry, e.g. `cf14d3a0-72ba-45f4-8814-3035d5608708`. Externalkeys are not guaranteed to be unique on the system.

Responses

200 — Successful operation

`name` and `description` are **multilingual objects**: keys are ISO 639-1 language codes (e.g. `de-ch`, `fr`) and values are the translated strings. Which languages are present depends on the data in the system — the API does not guarantee a fixed set of languages.

Field	Type	Required	Description
`id`	string	No[1]	Unique identifier (COO address) of the fileplan entry.
`externalkey`	string	No[1]	Externalkey of the fileplan entry.
`name`	object	No[1]	Translated name — see the multilingual note above.
`description`	object	No[1]	Translated description — see the multilingual note above.
`validfrom`	string (date-time)	No[2]	Start of the validity period. Plain ISO 8601 string with millisecond precision and an explicit UTC offset, e.g. `2000-01-23T04:56:07.000+00:00`. Can be `null`.

`validto`	string (date-time)	No[2]	End of the validity period. Same format as `validfrom`. Can be `null`.
`primaryrelated`	string	No[1]	COO address of the parent object (either a fileplan or fileplan entry).
`dossiers`	array of string	No[1]	COO addresses of child dossiers. Mutually exclusive with `fileplanentries` — a fileplan entry has either dossiers or child fileplan entries, not both.
`fileplanentries`	array of string	No[1]	COO addresses of child fileplan entries. Mutually exclusive with `dossiers`.
`businessnumber`	string	No[1]	Business number of the fileplan entry, e.g. `3.1`.
`location`	string	No[1]	Physical or organisational location of the fileplan entry, e.g. `Archiv Bern`.
`responsible`	object	No[1]	The object responsible for the fileplan entry — see the `responsible` table below.
`sequencenumber`	integer	No[1]	Current sequence number of the fileplan entry.
`classification`	string	No[1]	Classification of the fileplan entry. See the allowed values listed under **POST /fileplanentries → Request Body → `classification`**.
`privacyprotection`	boolean	No[1]	Whether the fileplan entry is privacy protected.
`disclosurestatus`	string	No[1]	Status of the disclosure. See the allowed values listed under **POST /fileplanentries → Request Body → `disclosurestatus`**.
`disclosurestatusstatement`	string	No[1]	Statement for the disclosure status.
`retentionperiod`	integer	No[1]	Retention period in years.
`retentionperiodcomment`	string	No[1]	Comment for the retention period.

`archivalvalue`	string	No[1]	Decision whether the fileplan entry is predestined to be archived. See the allowed values listed under **POST /fileplanentries → Request Body → `archivalvalue`**.
`archivalvaluecomment`	string	No[1]	Comment for the archival value.
`regularsafeguardperiod`	integer	No[1]	Time period (in years) for which the fileplan entry is retained in the archive.
`changedby`	string	No[1]	COO address of the user who last edited the fileplan entry.
`modifiedat`	string (date-time)	No[2]	Last modified timestamp. Same ISO 8601 format as `validfrom`, e.g. `2000-01-23T04:56:07.000+00:00`. Can be `null`.
`createdby`	string	No[1]	COO address of the user who created the fileplan entry.
`createdat`	string (date-time)	No[2]	Creation timestamp. Same ISO 8601 format as `validfrom`. Can be `null`.
`administrators`	array of string	No[1]	COO addresses of users with admin rights on the fileplan entry.
`changeaccess`	array of string	No[1]	COO addresses of users with change access.
`readaccess`	array of string	No[1]	COO addresses of users with read access.

[1] Not marked as required in the schema, but the server always populates this field. [2] Not marked as required in the schema; the server always includes this field, but its value can be `null`.

`responsible` has the following fields:

Field	Type	Required	Description
`name`	string	No[1]	Display name of the responsible object.
`id`	string	No[1]	COO address of the responsible object.

[1] Not marked as required in the schema, but the server always populates this field.

404 — No fileplan entry was found

Field	Type	Required	Description
`type`	string	No[1]	Identifier of the error, e.g. `invalid-request-body`.
`title`	string	No[1]	Short summary of the error.
`detail`	string	No[1]	Detailed explanation of the error.
`status`	integer	No[1]	HTTP status code, repeated in the body.

[1] Not marked as required in the schema, but the server always populates this field.

Python Example

Python example

```
import requests

fpe_id = "COO.1.30000.1.1234"  # COO address or externalkey of a fileplan entry
url = f"{BASE_URL}/{SERVICE_COO}/fileplanentries/{fpe_id}"

response = requests.get(url, auth=AUTH)
print(response.status_code)
print(response.json())
```

Java Example

Java example

```
import java.net.URI;
import java.net.http.*;
import java.util.Base64;

String auth = Base64.getEncoder().encodeToString((AUTH_USER + ":" +
AUTH_PASSWORD).getBytes());
String fpeId = "COO.1.30000.1.1234"; // COO address or externalkey of a fileplan
entry

HttpClient client = HttpClient.newHttpClient();
HttpRequest request = HttpRequest.newBuilder()
```

```
    .uri(URI.create(BASE_URL + "/" + SERVICE_COO + "/fileplanentries/" + fpeId))
    .header("Authorization", "Basic " + auth)
    .GET()
    .build();

HttpResponse<String> response = client.send(request,
HttpResponse.BodyHandlers.ofString());
System.out.println(response.statusCode());
System.out.println(response.body());
```

5.3.4 POST /fileplanentries/{fpeId}

Updates an existing fileplan entry.

Parameters

Name	In	Required	Type	Description
`fpeId` (COO address)	path	Yes	string	COO address of a fileplan entry, e.g. `COO.1.30000.1.1234`. COO addresses are always unique on the system.
`fpeId` (externalkey)	path	Yes	string	Externalkey of a fileplan entry, e.g. `cf14d3a0-72ba-45f4-8814-3035d5608708`. Externalkeys are not guaranteed to be unique on the system.

Request Body

The request body has the following field:

Field	Type	Required	Description
`name`	string	No	Title of the fileplan entry (room). Language is chosen by the requester's set Fabasphere language. Default value `__empty__` — if this field is omitted (or explicitly sent as `__empty__`), the server treats it as if no value was provided, and the fileplan entry's name is left unchanged.

Responses

204 — Successful operation

No response body.

400 — Invalid data in the request body

Field	Type	Required	Description
`type`	string	No[1]	Identifier of the error type, e.g. `invalid-request-body`.
`title`	string	No[1]	Short summary of the error.
`detail`	string	No[1]	Detailed explanation of the error.
`status`	integer	No[1]	HTTP status code, repeated in the body.
`errors`	array of error segments	No[1]	One entry per individual validation problem — see the `errors[]` table below.

[1] Not marked as required in the schema, but the server always populates this field.

Each entry of `errors[]` has the following fields:

Field	Type	Required	Description
`detail`	string	No[1]	Human-readable explanation of this specific validation error.
`pointer`	string	No[1]	JSON Pointer (e.g. `#/name`) identifying which request field the error refers to.

[1] Not marked as required in the schema, but the server always populates this field.

404 — No fileplan entry (room) was found

Same `GWSResponseError` structure as described above for this operation — see that table for the field explanations.

500 — Unexpected server-side error

Same `GWSResponseError` structure as described above for this operation — see that table for the field explanations.

Python Example

Python example

```
import requests

fpe_id = "COO.1.30000.1.1234"  # COO address or externalkey of a fileplan entry
(room)
url = f"{BASE_URL}/{SERVICE_COO}/fileplanentries/{fpe_id}"

response = requests.post(url, json={"name": "Gemeinderat (aktualisiert)"},
auth=AUTH)
print(response.status_code)
```

Java Example

Java example

```
import java.net.URI;
import java.net.http.*;
import java.util.Base64;

String auth = Base64.getEncoder().encodeToString((AUTH_USER + ":" +
AUTH_PASSWORD).getBytes());
String fpeId = "COO.1.30000.1.1234"; // COO address or externalkey of a fileplan
entry (room)
String body = """
    {"name": "Gemeinderat (aktualisiert)"}
    """;

HttpClient client = HttpClient.newHttpClient();
HttpRequest request = HttpRequest.newBuilder()
    .uri(URI.create(BASE_URL + "/" + SERVICE_COO + "/fileplanentries/" + fpeId))
    .header("Authorization", "Basic " + auth)
    .header("Content-Type", "application/json")
    .POST(HttpRequest.BodyPublishers.ofString(body))
    .build();

HttpResponse<String> response = client.send(request,
HttpResponse.BodyHandlers.ofString());
System.out.println(response.statusCode());
```

6 Round-Trip Example

This example fetches an existing fileplan entry, changes its name, submits the update, and re-fetches the entry to confirm the change took effect. Note the type difference between the two operations: the **GET** response's `name` is a multilingual object keyed by language code, while the **update** request's `name` is a single plain string — the language it is stored under is chosen by the requester's Fabasphere language setting, not by the caller.

Python Example

Python — round-trip example

```
import requests

fpe_id = "COO.1.30000.1.1234"  # COO address or externalkey of an existing
fileplan entry
url = f"{BASE_URL}/{SERVICE_COO}/fileplanentries/{fpe_id}"

# 1) Fetch current state
before = requests.get(url, auth=AUTH).json()
print("Name before:", before["name"])

# 2) Update the name (plain string; language is chosen by the requester's
Fabasphere language)
update_response = requests.post(url, json={"name": "Gemeinderat (aktualisiert)"},
auth=AUTH)
print("Update status:", update_response.status_code)

# 3) Re-fetch to confirm the change took effect
after = requests.get(url, auth=AUTH).json()
print("Name after:", after["name"])
```

Java Example

Java — round-trip example

```
import java.net.URI;
import java.net.http.*;
import java.util.Base64;

String auth = Base64.getEncoder().encodeToString((AUTH_USER + ":" +
AUTH_PASSWORD).getBytes());
String fpeId = "COO.1.30000.1.1234"; // COO address or externalkey of an existing
fileplan entry
String url = BASE_URL + "/" + SERVICE_COO + "/fileplanentries/" + fpeId;
HttpClient client = HttpClient.newHttpClient();

// 1) Fetch current state
HttpRequest getBefore = HttpRequest.newBuilder(URI.create(url))
    .header("Authorization", "Basic " + auth).GET().build();
```

```
HttpResponse<String> before = client.send(getBefore,
HttpResponse.BodyHandlers.ofString());
System.out.println("Name before: " + before.body());

// 2) Update the name (plain string; language is chosen by the requester's
Fabasphere language)
String updateBody = """
    {"name": "Gemeinderat (aktualisiert)"}
    """;
HttpRequest update = HttpRequest.newBuilder(URI.create(url))
    .header("Authorization", "Basic " + auth)
    .header("Content-Type", "application/json")
    .POST(HttpRequest.BodyPublishers.ofString(updateBody))
    .build();
HttpResponse<String> updateResponse = client.send(update,
HttpResponse.BodyHandlers.ofString());
System.out.println("Update status: " + updateResponse.statusCode());

// 3) Re-fetch to confirm the change took effect
HttpRequest getAfter = HttpRequest.newBuilder(URI.create(url))
    .header("Authorization", "Basic " + auth).GET().build();
HttpResponse<String> after = client.send(getAfter,
HttpResponse.BodyHandlers.ofString());
System.out.println("Name after: " + after.body());
```