univention
be open.

Self Disclosure API manual 1.0.0

Release 1.0.0

Univention GmbH

Mar 04, 2024

CONTENTS:

The *Self-disclosure API* is a component of the *Univention ID Broker* and allows *service providers* to retrieve information about the users of their *service*. Users typically are students and teachers in schools.

This manual addresses administrators and software developers at service providers, that want to provide a service to users like students and teachers. The ID Brokers delivers information about the users to the service provider.

To follow this document you need basic knowledge about the following concepts:

- UCS@school
- HTTP requests and codes
- JSON objects and their structure

This document covers the Self-disclosure *API* within the context of the ID Broker, the *routes* (page 7), and *data models and their structures* (page 13).

This document doesn't cover the onboarding of *school authorities*, that want to connect to the ID Broker. For further information about these topics, see ID Broker manual for school authorities[1].

[1] https://docs.software-univention.de/idbroker-school-authority-manual/index.html#introduction

CHAPTER

ONE

SELF-DISCLOSURE API IN CONTEXT OF ID BROKER

The *Univention ID Broker* eases the integration between identities of learners and teachers managed by *school authorities* or federal states and the various *service providers* for educational purposes with respect to the data protection regulations in Europe.

To use the *API* and request information about users like students and teachers, the service provider needs an authentication token. The user's browser receives the authentication token from the *Identity Provider (IDP)* of their *school authority* and passes it to the service provider. With the authentication the service provider can then request additional information about the user from the Self-disclosure API.

1.1 Sequence diagram

Figure 1.1 shows the complete flow of a user that logs in to a service through a school's portal website. The ID Broker provides user data through the *Self-disclosure API*. *Step 8* in Figure 1.1 uses the *Self-disclosure API*.

The *Self-disclosure API* only returns service provider specific pseudonyms instead of real personal information. Two different service providers receive different pseudonyms for the same real user.

Note: At the time of writing some user data isn't pseudonymized yet.

1.2 Authentication token

In *step 6* the *Univention ID Broker* responds with an authentication token for the user that wants to sign in. A token looks like the following example:


```
{
    "access_token":
↪"eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJuRHBjM2szTkFzZGc4YndJbGRZRnk5MjVQYWxBR2g5bDI
↪eyJleHAiOjE2NDM4MDE1ODQsImlhdCI6MTY0MzgwMTI4NCwiYXV0aF90aW1lIjoxNjQzNzk3MjkyLCJqdGkiOiIzNjk2OTJm
↪tip2fcB6it5EcMfUXRbEoZfvc9ofpBjbrGXQJHD6yXYyWnImGJsWBd8-
↪6wfukxwIu433sVCIFy5k0bYkyqO9OHDs5RZJ4o6FTN8EEy4rN1Ne0r4WR_3kAoJPxiQMrC9K-6_
↪1GgtsOQyAbQns74zVeBj2zis1JlcusSNyIDidQ6beqsVBQalfdgnzrhejvh3LnVjPOedNtrFurUdbZ-
↪JIsWsv3uwo_OX6ZrtvUCvBzNEFT85gmq1uPqFTk4stW7jpVZSc5-
↪UKMcFNjVsMCHTDRaTffIL2WiUVPWQh239M7Q4NSPN7JBDm_
↪pMBtowOBgVEAda9YeXDgJUlrqc2cYjdZGX1GQ6dGtApeVCiCOv6OpmhFQmT4-fKKmpqatc_
↪Aia5NinJ21PstJDRIXe4F8T43KYtfAwxjwAcMxGzyiuvdDphrbDDTNxBsUH7IFDUw1QCyuzUavus3r8vw0z9XOiNINq_
↪-BuXnQrC0fkPLhqzshFnZGFJ8vJO24Fn2vhkaSh3CKctfusSEB1SRA9Kl2icT5BRZ_bNeEvyXk-hzmZ_
↪NhCiVdOPfIw4i_Cfb8KRd6BW0Nk68BSmWdTV1AAFRF3x9UB5sEOZooLK4gYwkX0kEduZ-
↪aGxK6t2Tz45XLCR_vsTonKe7o6F5D47b-VVhm4N8PQHQgq28Bd8VCZlyTqipUU",
    "expires_in": 300,
    "refresh_expires_in": 1800,
    "refresh_token":
↪"eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICI3ZWEwMDYzNi0yZTBlLTQ0YzMtOGRhM$1kMGM1M2U5ZTc
```


(continues on next page)

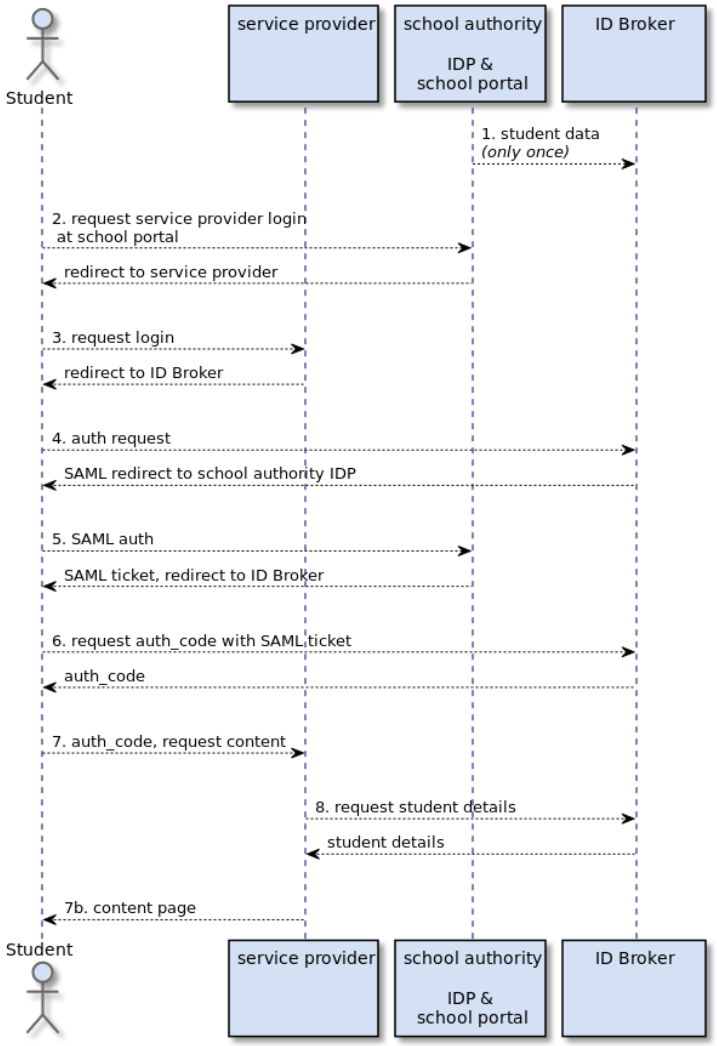

Figure 1.1: ID Broker sequence: authentication and user data retrieval sequence

(continued from previous page)


```
↪eyJleHAiOjE2NDM4MDMwODQsImlhdCI6MTY0MzgwMTI4NCwianRpIjoiMmZhNjdmZGUtNzUwNy00NDYzLTlhYzAtZGJmMDI
↪DkTZc2H_yqN8QJPV4YuUk1tqdbdpzCd56sCKZ7pDDQQ",
    "token_type": "Bearer",
    "id_token":
↪"eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJuRHBjM2szTkFzZGc4YndJbGRZRnk5MjVQYWxBR2g5bDI
↪eyJleHAiOjE2NDM4MDE1ODQsImlhdCI6MTY0MzgwMTI4NCwiYXV0aF90aW1lIjoxNjQzNzk3MjkyLCJqdGkiOiJhODBiZjhl
↪gVaJTVNwWwNRh04aat9BV8ObWaVuGZtSlOMybLktzwERjrG8TBJ8wpk9rWzaym7vl34QJ30cLmCLdLgHHXN_
↪rYpLiAkE6_L6sjfIwPqPHSCzMFCtohpnwdwHI-
↪tA0ii3eR6ms4EqutXNVrZNluR3ApI9GW8owGEit4SQQstBm_
↪qwYYMR5dfkAuIxh24g1U2NvGO5ZrtBU2o7zc0M26hgWQ6y9MAaMa-PYaVecK-u21d_MkWbccm2Kg-
↪j2ErNDX4N_1l6WELdZjppzSnH-zYY-TV7FbhV_jgS2FPoVt3SDOhsH6CBjxm4J5t48Vowjlf0vk0V6lm_
↪ys5batZ5Z1fvcd9-DcY8RjmHuvU_T5ocKrrK2dKcKNnjrnBZUiwQzoszjNY9t02HLmAI8mfL5nzyO-
↪OM6jX9YDa0jy_6QkeInQXN4Dgh2YNd333F9lPrnJph5oul2V9-
↪5lV6scD2hPRSwrnPVBAsVcW14WA2nttvYEQoJ-J637oXokP2XCqftcd5zYFV3mjzDEyuH0GMvkO8kpK6_
↪4s2hh9KFDzeIWTRoFfSbLhq5b_rpSasGzEb3tfR20YlCTAQo1j06iBsAkBdebopn8TKU_
↪eLFpExxc90pXOooNmznWGmZT36buIUQJIfiJh0C2WIWvU7au_gVKzU9cFuu54F2xUmQA97JEKsDCU",
    "not-before-policy": 0,
    "session_state": "134eae83-970a-4d3a-84a5-a8e331be7d22",
    "scope": "openid email profile"
}
```


The `acces_token`, `refresh_token` and the `id_token` are *JSON web tokens (JWT)* of the type *bearer*, also called JWT bearer tokens. The Self-disclosure API uses the token information to provide pseudonymized data to the service provider. Based on the data the service provider decides what content it presents to each user.

CHAPTER

TWO

ROUTES

The *Self-disclosure API* consists of multiple routes to retrieve data about users. The following sections explain the individual routes. You can find a corresponding *Swagger User Interface (Swagger UI)* at `https://FQDN/ucsschool/apis/docs#/` of your Self-disclosure API instance.

2.1 How to read

Each of the following sections makes use of a pattern. For better orientation, this section provides an overview of this pattern. Every URL follows a scheme.

Assume the URL is `https://FQDN/ucsschool/apis/self_disclosure/v1/users/0123456789`

You can decompose the URL into the parts in Table 2.1.

Table 2.1: Schema of the Self-disclosure API

Prefix	`https://FQDN`
Base route URL	`/ucsschool/apis/self_disclosure/v1`
Suffix	`/users/0123456789`

Description
: Provides a short summary of the requests that you can run against the specified route. The description offers critical information about the received objects.

Base route URL
: Provides the non-prefixed URL. The sections about the *requests* in each chapter provide suffixes to the specified URL.

Requests
: Provides a table per suffix that contains the parameters applicable for the resulting URL with additional information about each parameter and the needed HTTP-Method.

Examples
: Provides code examples for requests against the specified URL.

2.2 User metadata

Retrieve metadata about the requested user.

Note: Only people entitled to modify the metadata can retrieve metadata.

2.2.1 Requests

URL-suffix

`/users/id/metadata`

Method	Name	Type	Required	Possible values
GET	id	string	yes	

Composed URL:

```
FQDN =      # FQDN or IP-v4 address (string)
ID =        # specific user pseudonym (string)
metadata = f"https://{FQDN}/ucsschool/apis/self_disclosure/v1/users/{ID}/metadata"
```

2.2.2 Responses

A valid request returns a JSON object. For example:

```
{
  "user_id": "2849f8b0-9c98-4710-9fe1-23d4466g9af8",
  "username": "Max.Mustermann",
  "firstname": "Max",
  "lastname": "Mustermann",
  "type": "teacher",
  "school_id": "School1",
  "school_authority": "SchoolAuthority1"
}
```

The `user_id` field contains a unique pseudonym for each user. The `user_id` and the combination of `school_id` and `school_authority` are unique across all schools connected to the ID Broker. The other fields are not unique.

The field `type` refers to a UCS@school object class, that can have one of the following values: `teacher`, `student`, `admin`, `staff`, `teacher-staff`.

2.2.3 Examples

Curl

```
$ export HOST="[IP or FQDN]"
$ export ID="[ID]"
$ export TOKEN="[TOKEN]"

$ curl -X 'GET' \
> 'http://$HOST/ucsschool/apis/self_disclosure/v1/users/$ID/metadata' \
> -H 'accept: application/json' \
> -H 'Authorization: Bearer $TOKEN'
```

2.3 User specific group data

This route retrieves information about the group memberships of the requested user. It also retrieves additional information about the groups listed in the group membership.

2.3.1 Base route URL

```
/ucsschool/apis/self_disclosure/v1
```

2.3.2 Requests

URL-suffix:

`/users/{id}/groups`

Method	Name	Type	Required	Possible values
GET	id	string	yes	

Composed URL:

```
FQDN =         # FQDN or IP-v4 address (string)
ID =           # specific user pseudonym (string)
user_groups = f"https://{FQDN}/ucsschool/apis/self_disclosure/v1/users/{ID}/groups"
```

2.3.3 Responses

A valid request returns a JSON object. For example:

```
{
  "groups": [
    {
      "group_id": "503f1278-9cda-49f6-a749-3af022a7d32b",
      "name": "1a",
      "school_id", "School1",
      "school_authority": "SchoolAuthority1",
      "student_count": 2,
      "type": "school_class"
    }
  ]
}
```

The returned list shows all groups with their `group_id`, `name` and `student_count`. The field `type` refers to a UCS@school object class, that can have one of the following values: `school_class`, `workgroup`.

2.3.4 Examples

Curl

```
$ export HOST="[IP or FQDN]"
$ export ID="[ID]"
$ export TOKEN="[TOKEN]"

$ curl -X 'GET' \
> 'http://$HOST/ucsschool/apis/self_disclosure/v1/users/$ID/groups' \
> -H 'accept: application/json' \
> -H 'Authorization: Bearer $TOKEN'
```

2.4 Group members

Retrieve user data about the members of the requested group.

2.4.1 Base route URL

```
/ucsschool/apis/self_disclosure/v1
```

2.4.2 Requests

URL-suffix:

`/groups/{id}/users`

Method	Name	Type	Required	Possible values
GET	id	string	yes	

Composed URL:

```
FQDN =          # FQDN or IP-v4 address (string)Changed aspects:
ID =            # specific group pseudonym (string)
group_members = f"https://{FQDN}/ucsschool/apis/self_disclosure/v1/groups/{ID}/
↪users"
```

2.4.3 Responses

A valid request returns a JSON object. For example:

```
{
  "students": [
    {
      "user_id": "0e49f8b0-9c98-4716-9fe1-23d4466d9af8",
      "username": "Jane.Doe",
      "firstname": "Jane",
      "lastname": "Doe"
    },
    {
      "user_id": "86f753e0-50b3-44cb-807e-2c2556b0e6ca",
      "username": "John.Doe",
      "firstname": "John",
```

(continues on next page)

(continued from previous page)

```
      "lastname": "Doe"
    }
  ],
  "teachers": [
    {
      "user_id": "d8aa843b-9628-42b8-9637-d198c745efa5",
      "username": "Max.Mustermann"
    }
  ]
}
```

The response lists all members of the requested group, split by role (teacher, student).

2.4.4 Examples

Curl

```
$ export HOST="[IP or FQDN]"
$ export ID="[ID]"
$ export TOKEN="[TOKEN]"

$ curl -X 'GET' \
> 'http://$HOST/ucsschool/apis/self_disclosure/v1/groups/$ID/users' \
> -H 'accept: application/json' \
> -H 'Authorization: Bearer $TOKEN'
```

CHAPTER

THREE

SCHEMAS

This section covers the provided data classes that are present on a system with UCS@school installed. The responses to the requests covered in *section* (page 7) contain various possible combinations of the data classes.

The documentation format of the data classes is identical to the "Schemas"-section within the *Swagger User Interface (Swagger UI)* available at `https://FQDN/ucsschool/apis/docs#/`. Replace *FQDN* with the fully qualified domain name of your UCS@school system. You can consult this section, if you can't use the Swagger UI and you need information about data classes.

3.1 GroupMembers

```
GroupMembers {
  students [
  title: Students
    student {
      user_id*    string
                  title: User Id
      username*   string
                  title: Username
      firstname*  string
                  title: Firstname
      lastname*   string
                  title: Lastname
    }]
  teachers [
  title: Teachers
    teacher {
      user_id*    string
                  title: User Id
      username*   string
                  title: Username
    }]
}
```

3.2 CountedGroup

```
CountedGroup {
  group_id*         string
                    title: Group Id
  type*             string
                    title: Type
  name*             string
                    title: Name
  school_id*        string
```

(continues on next page)

(continued from previous page)

```
                    title: School Id
  school_authority* string
                    title: School Authority
  student_count*    integer
                    title: Student Count
}
```

3.3 HTTPValidationError

```
HTTPValidationError {
  detail [
  title: Detail
    ValidationError {
      loc*        Location [
                  title: Location
                    string
                  ]
      msg*        string
                  title: Message
      typ*        string
                  title: Error Type

    }]
}
```

3.4 Student

```
Student {
  user_id*    string
              title: User Id
  username*   string
              title: Username
  firstname*  string
              title: Firstname
  lastname*   string
              title: Lastname
}
```

3.5 Teacher

```
teacher {
  user_id*    string
              title: User Id
  username*   string
              title: Username
}
```

3.6 Token

```
Token {
  access_token*    string
                   title: Access Token
  token_type*      string
                   title: Token Type
}
```

3.7 User

```
User {
  user_id*          string
                    title: User Id
  username*         string
                    title: Username
  firstname*        stringLastname
                    title: Firstname
  lastname*         string
                    title: Lastname
  type*             string
                    title: Type
  school_id*        string
                    title: School Id
  school_authority* string
                    title: School Authority
}
```

3.8 UsersGroups

```
UsersGroups {
  groups [
  title: Groups
    CountedGroup {
    group_id*         string
                      title: Group Id
    type*             string
                      title: Type
    name*             string
                      title: Name
    school_id*        string
                      title: School Id
    school_authority* string
                      title: School Authority
    student_count*    integer
                      title: Student Count
    }]
}
```

3.9 ValidationError

```
ValidationError {
  loc*        Location [
              title: Location
                  string
              ]
  msg*        string
              title: Message
  typ*        string
              title: Error Type
}
```

CHAPTER

FOUR

GLOSSARY

JSON Web Token (JWT)
A JWT is standardized way to securely transmit information between parties in a compact and self-contained manner. For the specification, see **RFC 7519**[2].

Application Programming Interface (API)
Is a set of functions and procedures that allow users to access features and data from the corresponding system.

Swagger User Interface (Swagger UI)
Allows visual interaction with the API's resources without the need for an external implementation. You find the URLs to the respective Swagger UI in the appropriate sections.

Identity Provider (IDP)
Instance that provides information to authenticate and authorize identities. In case of ID Broker scenarios this is typically a SAML or OpenID Connect IDP hosted by a *School Authority*.

School Authority
In context of this document, the term *school authority* subsumes various institutions which serve one or several schools with IT infrastructure. The school authority is the data source for all students and teachers of an environment. The ID Broker will receive a minimal subset of this data, see *ID Broker sequence: authentication and user data retrieval sequence* (page 4). This can be a single school, a school authority with several schools, or an environment hosting services for a federal state. The environments are hosting a UCS@school domain.

Service
In the context of this document a *service* is an application, which uses single sign-on with the ID Broker and provides a service for students and teachers. For example a learning platform, that offers books.

Service Provider (SP)
Instance that provides a *service*.

[2] https://datatracker.ietf.org/doc/html/rfc7519.html

INDEX