Edulog

educa.ch

Edulog API reference

Project nr.	Report nr.	Version	Date	Author	Status	Visa
17691	-	1.4	17.01.2022	MBI/LPA	Final	

Table of Contents

1 Introduction 4

1.1 Environments 4

1.1.1 Integration 4

1.1.2 Production 4

1.2 API protocols 4

1.2.1 Proprietary API 4

1.2.2 SCIM API 4

1.3 URL domain 5

1.4 Attribute names 5

2 Authentication 6

2.1 Obtaining an access token 6

2.1.1 Request 6

2.1.2 Reply 6

2.1.3 Error return codes 7

2.1.4 Example 7

2.2 Refreshing the access token 7

2.2.1 Request 8

2.2.2 Reply 8

2.2.3 Error return codes 8

2.2.4 Example 8

2.3 SCIM: long-lived token 9

2.3.1 Request an offline token 9

2.3.2 Reply 10

2.3.3 Error return codes 10

2.3.4 Example 10

3 Proprietary API 12

3.1 Federate identity operation 12

3.1.1 Prerequisite 12

3.1.2 Request 12

3.1.3 Reply 12

3.1.4 Error return codes 12

3.1.5 Example 13

3.2 De-federate identity operation 13

3.2.1 Prerequisite 13

3.2.2 Request 13

3.2.3 Reply 13

3.2.4 Error return codes ... 14
3.2.5 Example ... 14
3.3 Update identity ... 14

4 SCIM API ... 15
4.1 Prerequisite ... 15
4.2 User representation ... 15
4.3 SCIM endpoints ... 16
4.4 Create identity operation ... 16
4.4.1 Request ... 16
4.4.2 Reply (success) ... 16
4.5 Update identity operation ... 17
4.5.1 Request for changing the userName ... 17
4.5.2 Reply (success) ... 17
4.5.3 Request for disabling the user ... 18
4.5.4 Reply (success) ... 18
4.6 Delete identity operation ... 19
4.6.1 Request ... 19
4.6.2 Reply (success) ... 19
4.7 List identities operation ... 20
4.7.1 Request (without pagination) ... 20
4.7.2 Reply (success) ... 20
4.7.3 Request (with pagination) ... 21
4.7.4 Reply (success) ... 21
4.8 Query identities operation ... 22
4.8.1 Request ... 22
4.8.2 Reply (success) ... 22
4.9 Get identity operation ... 23
4.9.1 Request ... 23
4.9.2 Reply (success) ... 23

Record of changes ... 24

Reference documents ... 24

Abbreviations ... 24

1 Introduction

This document gives technical details about the Edulog API. This API follows the REST principles and must be used by Identity Providers (IdP) to federate their identities, as explained in [onboarding-pptx].

1.1 Environments

Edulog offers 2 environments that are completely separated.

1.1.1 Integration

This environment is available if you want to test your integration with Edulog before going to production.

This environment has the same features as the production one, with the following limitations:

1. A new version of the API may be deployed at any time without communication
2. Your data may be deleted if necessary, to free up resources for newcomers. This will not be done before you are connected to production

Connecting your IdP to the INTEGRATION environment is optional.

1.1.2 Production

This is the real Edulog environment.

1.2 API protocols

Edulog offers two different API versions.

1.2.1 Proprietary API

This API is the original one provided by Edulog. It is “proprietary” in the sense that its protocol does not match any official standard.

It offers two simple methods to federate and defederate your identities.

This version of the API is described in details in § 3 “Proprietary API”.

1.2.2 SCIM API

Edulog also offers another API that follows the SCIM protocol. This standard is defined in RFC-7642, RFC-7643 and RFC-7644.

Some IT solutions from the market have built-in SCIM connectors, so you may use this one if it simplifies your deployment.

This version of the API is described in details in § 4 “SCIM API”.

1.3 URL domain

In this document, all URLs are prefixed with “AUTHDOMAIN” or “APIDOMAIN”. You must replace this value with the real DNS sub-domain used by Edulog.

This value is different depending on the environment you want to connect to:

	Integration	**Production**
AUTHDOMAIN	https://go.int.edulog.ch	https://go.edulog.ch
APIDOMAIN	https://api.int.edulog.ch	https://api.edulog.ch

1.4 Attribute names

This document uses attribute names specified in [Edulog-attributes].

More precisely, the following attributes are used by the API:

- **EdulogPersonTechId (section 5.12)**
  The unique identifier of the user in Edulog. It never changes.

  It is a UUID that looks like this:
  110e8400-e29b-11d4-a716-446655440000

- **uid (section 5.13)**
  The unique identifier of the user in your system.

  Important note: from an IAM perspective, it is a good idea to use a “uid” that never changes, such as a contract number, a student ID, etc.
  However, users need to provide their “uid” the first time they connect to Edulog; to retrieve their pseudonym.
  Therefore users must know their “uid”, so in practice a more user-friendly attribute such as an email address should be used (note that Edulog supports modifying the uid as explained in §3.3).

- **ahvn13 (not part of [Edulog-attributes])**
  The unique identifier of the physical person.

  This needs to be a valid AHVn13 number (with the dots and the correct checksum digit).
  For people that do not have a AHVn13, then the special value “999.9999.9999.99” can be used. Note that every time such user is federated, he will receive a new profile in Edulog (i.e. new techID, new pseudonym). As this is not user friendly at all, this special value must ONLY be used when the real AHVn13 does not exist.

2 Authentication

To be allowed to call the Edulog API methods, you must be authenticated. This is done using the OAuth protocol ("Resource Owner Password Credentials" grant type). The overall flow is the following:

1. You authenticate to Edulog token endpoint using your credentials
2. The token endpoint sends you back an access token and a refresh token
3. You call the Edulog API with this access token as a bearer token

2.1 Obtaining an access token

2.1.1 Request

To get your access token, you send an HTTP POST request to the token endpoint available at:

```
https://AUTHDOMAIN/auth/realms/edulog/protocol/openid-connect/token
```

The body of your request must be the following (no line break):

```
grant_type=password&client_id=federation&username=myUsername
&password=myPassword
```

In more technical words, this correspond to a POST request with the following parameters sent as *application/x-www-form-urlencoded*:

Name	Value
grant_type	password
client_id	federation
username	*myUsername*
password	*myPassword*

Replace ***myUsername*** and ***myPassword*** with your real credentials that are provided by ELCA during your onboarding process.

NOTE: these credentials are personal and must be kept secret. They must not be shared with anyone. In case several people need access to this API, several accounts must be used. Additional account can be requested within the limits of reasonable demands.

2.1.2 Reply

If authentication is successful, Edulog replies with an HTTP-200.

The body of this reply is a JSON object, that contains in particular an "access_token" attribute that you can use to authenticate to the API (see example below). This access token is a long base64 string, that represents a JWT token that you can parse with any compatible library. Thus, you can easily determine the expiration time of the token.

2.1.3 Error return codes

Specific codes returned by the API are documented in the Swagger.

2.1.4 Example

2.1.4.1 Request

```
POST https://AUTHDOMAIN/auth/realms/edulog/protocol/openid-
connect/token
Accept: */*
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 81

grant_type=password&client_id=federation&username=edulolgUsername&p
assword=edulogPassword
```

2.1.4.2 Reply

```
HTTP/1.1 200 OK
Cache-Control: no-store
Pragma: no-cache
Content-Type: application/json
Content-Length: 2323

{
    "access_token":"eyJhbGciOiJSUzI1NiIsInR-truncated-for-brevity”,
    "expires_in":300,
    "refresh_expires_in":1800,
    "refresh_token":"eyJhbGciOiJIUzI1NiInR5-truncated-for-brevity",
    "token_type":"bearer",
    "not-before-policy":0,
    "session_state":"a9e019ea-2411-4679-b35a-674e0c4c4c89",
    "scope":"openid groups email profile"}
}
```

2.2 Refreshing the access token

The access token has a limited validity. If it is used once it has expired, then the API will return a 401 error. In this case, you need to get a new access token using the refresh token that you retrieved during authentication.

Note that the refresh token has a longer validity than the access token, but once it has expired, you need to authenticate again (with username and password, as explained in chapter 2) in order to get a new one.

2.2.1 Request

To get your access token, you send an HTTP POST request to the token endpoint available at:

```
https://AUTHDOMAIN/auth/realms/edulog/protocol/openid-connect/token
```

The body of your request must be the following (no line break):

```
grant_type=refresh_token&refresh_token=refreshToken
&client_id=federation
```

In more technical words, this correspond to a POST request with the following parameters sent as *application/x-www-form-urlencoded*:

Name	Value
grant_type	refresh_token
client_id	federation
refresh_token	***refreshToken***

Replace ***refreshToken*** with the refresh token that was returned during authentication.

2.2.2 Reply

If refresh is successful, Edulog replies with an HTTP-200.

The body of this reply is a JSON object, that contains a new “access_token” attribute that you can use to authenticate to the API (see example below).

2.2.3 Error return codes

Specific codes returned by the API are documented in the Swagger.

2.2.4 Example

2.2.4.1 Request

```
POST https://AUTHDOMAIN/auth/realms/edulog/protocol/openid-
connect/token
Accept: */*
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 850

grant_type=refresh_token&refresh_token=eyJhbGcibXUK-truncated-for-
brevity&client_id=federation
```

2.2.4.2 Reply

```
HTTP/1.1 200 OK
Cache-Control: no-store
Pragma: no-cache
Content-Type: application/json
Content-Length: 3217

{
    "access_token":"eyJhbGciOiJSUzI1NiIsI-truncated-for-brevity",
    "expires_in":300,
    "refresh_expires_in":1800,
    "refresh_token":"eyJhbGciOiJIUzI1NiIs-truncated-for-brevity",
    "token_type":"bearer",
    "not-before-policy":0,
    "session_state":"a9e019ea-2411-4679-b35a-674e0c4c4c89",
    "scope":"openid groups email profile"
}
```

2.3 SCIM: long-lived token

For the SCIM interface, alternatively to an access token, there is the possibility to use a long-lived token. This token is also called “offline token” and can be used to authenticate on the SCIM interface.

Offline tokens have a much longer validity period (up to 400 days). Once it has expired, a new token must be obtained.

2.3.1 Request an offline token

To get an offline token, you send an HTTP POST request to the token endpoint available at:

`https://AUTHDOMAIN/auth/realms/edulog/protocol/openid-connect/token`

The body of your request must be the following (no line break):

```
grant_type=password&client_id=federation&username=myUsername
&password=myPassword&scope=offline_access
```

In more technical words, this correspond to a POST request with the following parameters sent as *application/x-www-form-urlencoded*:

Name	Value
grant_type	password
client_id	federation
username	*myUsername*
password	*myPassword*
scope	*offline_access*

Replace ***myUsername*** and ***myPassword*** with your real credentials that are provided by ELCA during your onboarding process.

NOTE: these credentials are personal and must be kept secret. They must not be shared with anyone.

2.3.2 Reply

If authentication is successful, Edulog replies with an HTTP-200.

The body of this reply is a JSON object, that **contains in particular an “refresh_token”** attribute that you can use to authenticate to the SCIM API (see example below)

2.3.3 Error return codes

Specific codes returned by the API are documented in the Swagger.

2.3.4 Example

2.3.4.1 Request

```
POST https://AUTHDOMAIN/auth/realms/edulog/protocol/openid-
connect/token
Accept: */*
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 81

grant_type=password&client_id=federation&username=edulolgUsername&
password=edulogPassword&scope=offline_access
```

2.3.4.2 Reply

```
HTTP/1.1 200 OK
Cache-Control: no-store
Pragma: no-cache
Content-Type: application/json
Content-Length: 2323

{
       "access_token": "eyJhbGciOiJSUzI1Ni-truncated-for-brevity",
       "expires_in": 60,
       "refresh_expires_in": 0,
       "refresh_token": "eyJhbGciOiJIUzI-truncated-for-brevity",
       "token_type": "bearer",
       "not-before-policy": 1620633352,
       "session_state": "b19687cb-8feb-43bb-8913-7527c2c13157",
       "scope": "offline_access groups"
}
```

3 Proprietary API

3.1 Federate identity operation

The federation endpoint enables you to register a user in Edulog. Once federated, your user can connect to Edulog.

You must call this endpoint once for each user. That means if you have 10’000 identities to federate to Edulog, you have to call the endpoint 10’000 times.

3.1.1 Prerequisite

You need an access token, as explained in chapter 2.

3.1.2 Request

You send an HTTP POST request to the federation endpoint available at:

`https://APIDOMAIN/federate/realms/edulog/federation/users`

The header of your request must contain an “Authorization” header with your access token:

```
Authorization: Bearer the-access-token-retrieved-from-the-authZ-endpoint
```

The body of your request must be the following JSON object:

```
{
    "ahvn13":"756.1234.5678.90",
    "userIdentifier":"your-uid"
}
```

Replace:

- ***756.1234.5678.90*** with the AHVn13 (numéro AVS à 13 chiffres) of your user. See §1.4 for additional information about this parameter, including a special value you may use.

  Note that the format of the AHVn13 number is checked, and the checksum is verified (otherwise you shall receive a HTTP 400 error).
- ***your-uid*** with the uid of your user

3.1.3 Reply

If federation process is successful, Edulog replies with an HTTP-200.

The body of this reply is a JSON object, that contains a “techId” attribute, i.e. the EdulogPersonTechID assigned to the user that was federated.

3.1.4 Error return codes

Specific codes returned by the API are documented in the Swagger.

3.1.5 Example

3.1.5.1 Request

```
POST https://APIDOMAIN/federate/realms/edulog/federation/users
Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldU-
truncated-for-brevity
Content-Type: application/json
Content-Length: 64

{
    "ahvn13":"756.1234.5678.97",
    "userIdentifier":"john@idp.example.com"
}
```

3.1.5.2 Reply

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 53

{
    "techId": "110e8400-e29b-11d4-a716-446655440000"
}
```

3.2 De-federate identity operation

The de-federate endpoint allows you to remove a user from Edulog once he leaves your organisation.

3.2.1 Prerequisite

You need an access token, as explained in chapter 2.

3.2.2 Request

You send an HTTP DELETE request to the endpoint of your user, available at:

`https://APIDOMAIN/federate/realms/edulog/federation/users/`***`EdulogPersonTechID`***

Replace ***EdulogPersonTechID*** with the real ID of your user.

The header of your request must contain an “Authorization” header with your access token:

```
Authorization: Bearer the-access-token-retrieved-from-the-authZ-endpoint
```

The body of this request is empty (the ID is passed in the URL).

3.2.3 Reply

If de-federation is successful, this method sends an HTTP-200.

The body is empty.

3.2.4 Error return codes

Specific codes returned by the API are documented in the Swagger.

3.2.5 Example

3.2.5.1 Request

```
DELETE
https://APIDOMAIN/federate/realms/edulog/federation/users/110e8400-
e29b-11d4-a716-446655440000
Authorization: Bearer eyJhbGciOiJSUzI1NiI-truncated-for-brevity
```

3.2.5.2 Reply

```
HTTP/1.1 200 OK
```

3.3 Update identity

Whenever the uid of one of your user changes, you must update his/her identity in Edulog.

Edulog does not provide an API to update a uid directly. Instead, the procedure is:

1. Call the “defederate” endpoint (cf chapter 3.2) to remove the old uid
2. Then call the “federate” endpoint (cf chapter3.1) to register the new one

4 SCIM API

The SCIM interface offers the same federate / defederate operations as the proprietary API but structures the payload according to the SCIM protocol[1].

This section gives an overview of the operations exposed through the SCIM protocol. For details about the data in the payloads, please refer to the equivalent method in the proprietary API (§ 3).

4.1 Prerequisite

For each of the calls below, you need an access token or an offline token, as explained in chapter 2.

The header of your request must contain an “Authorization” header with your access / offline token:

```
Authorization: Bearer the-token-retrieved-from-the-authZ-endpoint
```

4.2 User representation

The user attributes are mapped to the SCIM attributes as follows:

User attribute	SCIM attribute	SCIM schema
techID	id	urn:ietf:params:scim:schemas:core:2.0:User
userIdentifier	userName	urn:ietf:params:scim:schemas:core:2.0:User
ahvn13	ahvn13	urn:ietf:params:scim:schemas:extension:Edulog:2.0:User

Note that all Edulog users are “active”. Inactive users are defederated.

```
{
   "schemas": [
      "urn:ietf:params:scim:schemas:core:2.0:User"
   ],
   "id": string that represents the EdulogPersonTechID,
   "userName": string that represents the uid,
   "active": true,
   "urn:ietf:params:scim:schemas:extension:Edulog:2.0:User": {
      "ahvn13": unique string representing the ahvn13 number
   }
   "meta": {
      "resourceType": "User",
      "created": "2021-01-23T04:56:22Z"
   }
}
```

[1] https://en.wikipedia.org/wiki/System_for_Cross-domain_Identity_Management

4.3 SCIM endpoints

These standard SCIM endpoints describe the features and schemas used for this SCIM interfaces:

Resources:	https://APIDOMAIN/scim/v2/realms/edulog/ResourceTypes
Schemas:	https://APIDOMAIN/scim/v2/realms/edulog/Schemas
ServiceProviderConfig:	https://APIDOMAIN/scim/v2/realms/edulog/ServiceProviderConfig

4.4 Create identity operation

This operation is equivalent to federate identity (§ 3.1).

4.4.1 Request

```
POST /scim/v2/realms/edulog/Users
Content-Type: application/scim+json
Authorization: Bearer eyJhbGciOiJSUzI1NiIsI...
Host: APIDOMAIN

{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User",
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User"
  ],
  "userName": "max.muster@school.ch",
  "meta": {
    "resourceType": "User"
  },
  "urn:ietf:params:scim:schemas:extension:Edulog:2.0:User": {
    "ahvn13": "756.1234.5678.97"
  }
}
```

4.4.2 Reply (success)

The EdulogPersonTechId is returned by the API in the *id* attribute

```
HTTP/1.1 201 Created
Content-Type: application/scim+json; charset=utf-8
Content-Length: 237

{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User"
  ],
  "id": "5ca28bef-e599-497f-9ae8-83ae4aef753a",  ← this is the techID
  "userName": "max.muster@school.ch",
  "active": true,
  "meta": {
    "resourceType": "user",
    "created": "2021-05-18 08:43:16 +0000 UTC"
  }
}
```

4.5 Update identity operation

This operation allows to modify the userName (uid) and the active flag of a given user. All other requested changes are ignored.

The EdulogPersonTechId needs to be specified in the query URL, as a path parameter.

4.5.1 Request for changing the userName

```
PATCH /scim/v2/realms/edulog/Users/5ca28bef-e599-497f-9ae8-83ae4aef753a
Content-Type: application/scim+json
Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5c...
Host: APIDOMAIN

{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:PatchOp"
  ],
  "Operations": [
    {
      "op": "Replace",
      "path": "userName",
      "value": "max.muster2@school.ch"
    }
  ]
}
```

4.5.2 Reply (success)

The EdulogPersonTechId is returned by the API in the *id* attribute

```
HTTP/1.1 200 OK
Content-Type: application/scim+json; charset=utf-8
Content-Length: 363

{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User"
  ],
  "id": "5ca28bef-e599-497f-9ae8-83ae4aef753a",
  "userName": "max.muster2@school.ch",
  "active": true,
  "meta": {
    "resourceType": "user",
    "created": "2021-05-18 08:43:16 +0000 UTC"
  }
}
```

4.5.3 Request for disabling the user

Note that for Edulog, *disabling a user is equivalent to defederating the user*.

```
PATCH /scim/v2/realms/edulog/Users/5ca28bef-e599-497f-9ae8-83ae4aef753a
Content-Type: application/scim+json
Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5c...
Host: APIDOMAIN

{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:PatchOp"
  ],
  "Operations": [
    {
      "op": "Replace",
      "path": "active",
      "value": false
    }
  ]
}
```

4.5.4 Reply (success)

The reply after disabling of a user is an empty response.

```
HTTP/1.1 204 No Content
Content-Type: application/scim+json; charset=utf-8
```

4.6 Delete identity operation

This operation is equivalent to defederate identity (§ 3.2).

4.6.1 Request

The EdulogPersonTechId need to be specified in the query URL, as a path parameter.

```
DELETE /scim/v2/realms/edulog/Users/5ca28bef-e599-497f-9ae8-83ae4aef753a
Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCIgOi...
Host: APIDOMAIN
```

4.6.2 Reply (success)

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

4.7 List identities operation

This operation is implemented for SCIM compliance. It lists all users currently federated.

This operation supports pagination. The following query parameters can be specified:

- startIndex: 1-based index of the first query result
- count: desired maximum number of query results per page

4.7.1 Request (without pagination)

```
GET /scim/v2/realms/edulog/Users
Authorization: Bearer eyJhbGciOiJSUzI1N...
Host: APIDOMAIN
```

4.7.2 Reply (success)

```
HTTP/1.1 200 OK
Content-Type: application/scim+json; charset=utf-8

{
 "schemas": [
  "urn:ietf:params:scim:api:messages:2.0:ListResponse"
 ],
 "totalResults": 8,
 "Resources": [
  {
   "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User"
   ],
   "id": "e04877ea-8ca7-45a2-bc74-8fab85af4309",
   "userName": "cynthia.jotterand",
   "active": true,
   "meta": {
    "resourceType": "user",
    "created": "2020-05-08 11:32:47 +0000 UTC"
   }
  },
  ...
 ],
 "startIndex": 1,
 "itemsPerPage": 100
}
```

4.7.3 Request (with pagination)

```
GET /scim/v2/realms/edulog/Users?count=1&startIndex=2
Authorization: Bearer eyJhbGciOiJSUzI...
Host: APIDOMAIN
```

4.7.4 Reply (success)

```
HTTP/1.1 200 OK
Content-Type: application/scim+json; charset=utf-8

{
 "schemas": [
  "urn:ietf:params:scim:api:messages:2.0:ListResponse"
 ],
 "totalResults": 8,
 "Resources": [
  {
   "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User"
   ],
   "id": "a8381734-7206-45be-ae42-59c295968bc8",
   "userName": "hans.mueller",
   "active": true,
   "meta": {
    "resourceType": "user",
    "created": "2020-05-11 15:15:08 +0000 UTC"
   }
  }
 ],
 "startIndex": 2,
 "itemsPerPage": 1
}
```

4.8 Query identities operation

This operation is implemented for SCIM compliance. The current implementation is limited to query some specific user by userName, with exact equality.

4.8.1 Request

```
GET /scim/v2/realms/edulog/Users?filter=userName eq "t.clark"
Authorization: Bearer eyJhbGciOiJSUzI1NiIsI...
Host: APIDOMAIN
```

4.8.2 Reply (success)

```
HTTP/1.1 200 OK
Content-Type: application/scim+json; charset=utf-8

{
 "schemas": [
  "urn:ietf:params:scim:api:messages:2.0:ListResponse"
 ],
 "totalResults": 1,
 "Resources": [
  {
   "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User"
   ],
   "id": "4968cdfa-7b48-4ad1-9b6e-8c9c16012bf6",
   "userName": "t.clark",
   "active": true,
   "meta": {
    "resourceType": "user",
    "created": "2020-05-09 11:35:57 +0000 UTC"
   }
  }
 ],
 "startIndex": 1,
 "itemsPerPage": 100
}
```

4.9 Get identity operation

This operation is implemented for SCIM compliance. It retrieves a single specific user based on the EdulogPersonTechId.

4.9.1 Request

```
GET /scim/v2/realms/edulog/Users/e04877ea-8ca7-45a2-bc74-8fab85af4309
Authorization: Bearer eyJhbGciOiJSUzI1NiIsIn...
Host: APIDOMAIN
```

4.9.2 Reply (success)

```
HTTP/1.1 200 OK
Content-Type: application/scim+json; charset=utf-8
Content-Length: 234

{
 "schemas": [
  "urn:ietf:params:scim:schemas:core:2.0:User"
 ],
 "id": "e04877ea-8ca7-45a2-bc74-8fab85af4309",
 "userName": "cynthia.jotterand",
 "active": true,
 "meta": {
  "resourceType": "user",
  "created": "2020-05-08 11:32:47 +0000 UTC"
 }
}
```

Record of changes

Filename	Version	Date	Description / Author
Edulog-API-reference-1.0.docx	1.0	22.05.2020	First version / MBI/LPA
Edulog-API-reference-1.1.docx	1.1	08.06.2020	Add refresh token and SHVn13 format / MBI/LPA
Edulog-API-reference-1.2.docx	1.2	18.05.2021	- add SCIM APIs - add offline token for SCIM / MBI/LPA
Edulog-API-reference-1.3.docx	1.3	02.06.2021	- add SCIM Update operation / LPA
Edulog-API-reference-1.4.docx	1.4	17.01.2022	- add active flag in SCIM API / LPA

Reference documents

onboarding-pptx	Edulog-onboarding-document.pptx
Edulog-attributes	Leitfaden zu den Attributen für Identitätsanbieter (IdP) Guide des attributs pour fournisseurs d'identité (IdP) (Guides available on https://edulog.ch => Beitritt/Adhésion => Dokumentation/Documentation)

Abbreviations

JWT	JSON Web Token
SCIM	System for Cross-domain Identity Management