Contents

Chapter 1

API: user-auth-jsonrpc /auth

This document describes the API for each method implemented in the `user-auth-jsonrpc` service. This is merely a documentation for the consumer of the service.

This documents includes a brief introduction about the JSON-RPC protocol on which every service is implemented. The clients of the service must also obey this protocol in order to successfully comunicate with the service.

Some of the method are protected by an API key. That API key must be defined throught the evironment variable `API_KEY`.

It is recommended to create an admin account the first time the service is run in order to perform admin tasks. The credential for this first admin account must be specified using the environment variables:

- `ADMIN_USER`: the email of the account
- `ADMIN_PASSWORD`: the password of the account

The account is created if it does not exist and it is not needed to confirm the register process.

You need to set the variables for the first admin account only once, and be sure to store it in a secure place.

Chapter 2

JSON-RPC Protocol

Specification

JSON-RPC is a stateless, light-weight remote procedure call (RPC) protocol. Primarily this specification defines several data structures and the rules around their processing. It is transport agnostic in that the concepts can be used within the same process, over sockets, over http, or in many various message passing environments. It uses JSON (RFC 4627) as data format.

The full specification can be found at http://www.jsonrpc.org/. In our implementation we are adopting the JSON-RPC 2.0 specification, see http://www.jsonrpc.org/specification

Transport

We are using the http transport to implement the JSON-RPC in which each service implement a single end-point wating for a POST request.

The content type for the request and the response is "application/json". The authorization is implemented based on a bearer token so the `Authorization: Bearer` http header is required.

The body of the request contains the json for the JSON-RPC request. The service should always return status code 200, other status codes returned must considered as unexpected errors. In case of success (code=200) the body of the response contains the json with result or error as specified by the JSON-RPC standard.

Examples

HTTP Call

```
http  -v POST http://localhost:40685/auth \
  'Authorization: Basic dXNlci10ZXN0QGdtYWlsLmNvbTpwYXNzd29yZA==' \
  <<< '{"jsonrpc": "2.0", "method":"login", "id":0}'
```

HTTP Request

```
POST /auth HTTP/1.1
Accept: application/json
Accept-Encoding: gzip, deflate
```

```
Authorization:  Basic dXNlci1OZXN0QGdtYWlsLmNvbTpwYXNzd29yZA==
Connection: keep-alive
Content-Length: 45
Content-Type: application/json
Host: localhost:40685
User-Agent: HTTPie/0.9.2

{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "login"
}
```

HTTP Response

```
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 442
Content-Type: application/json; charset=utf-8
Date: Wed, 22 Nov 2017 17:06:07 GMT
ETag: W/"1ba-ybuEi8ATVr+lQ9dgR4Pctn2ruqg"
X-Powered-By: Express

{
    "id": 0,
    "jsonrpc": "2.0",
    "result": {
        "email": "user-test@gmail.com",
        "token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IkdOM1JUdXFXZEtfQ3ZscWFHUmEyOHdFdmJTRnByUHNrNm9TMWNYdGl5N
    }
}
```

JSON-RPC Response


```
{
    "id": 0,
    "jsonrpc": "2.0",
    "result": {
        "email": "user-test@gmail.com",
        "token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IkdOM1JUdXFXZEtfQ3ZscWFHUmEyOHdFdmJTRnByUHNrNm9TMWNYdGl5N
    }
}
```

Chapter 3

API: register

The method `register` create a new user account given email and profile data for the user.

Request

path: `POST /auth`

method: `register`

The body must be a valid JSON-RPC request, with the following parameters:

- `email`: the user email, will be considered as the user identifier
- `password`: the user password
- `profile`: the user profile, must be a non empty object. In next release this object will be validated with respected to a json scheme.

Example

```
{
  "jsonrpc": "2.0",
  "id": "a9368c30-d2d9-11e7-864c-9fd13ae41559",
  "method": "register",
  "params": {
    "email": "user-test@gmail.com",
    "password": "password",
    "profile": {
      "name": "Paco",
      "surname": "Perico",
      "company": "Vago"
    }
  }
}
```

Response

In case of success an email is sent to the user with a confirmation url and a JSON-RPC result is returned. The result contains the email of the user registered.


```
{
  "jsonrpc": "2.0",
  "id": "a9368c30-d2d9-11e7-864c-9fd13ae41559",
  "result": {
    "email": "user-test@gmail.com"
  }
}
```


Errors

The following error responses can be generated:

- **if a parameter is missing**


```
{
  "jsonrpc": "2.0",
  "id": "79b35250-d2dd-11e7-bf97-e5a10b620ed7",
  "error": {
    "message": "Invalid params",
    "code": -32602,
    "data": {
      "message": "missing parameter",
      "parameter": "email"
    }
  }
}
```


- **if a parameter has an invalid value**


```
{
  "jsonrpc": "2.0",
  "id": "17443c50-d2de-11e7-88b3-5f372623f503",
  "error": {
    "message": "Invalid params",
    "code": -32602,
    "data": {
      "message": "parameter profile must be a non empty object",
      "parameter": "profile"
    }
  }
}
```


- **if the user is already registered**


```
{
  "jsonrpc": "2.0",
  "id": "1802d160-d2de-11e7-88b3-5f372623f503",
  "error": {
    "message": "Entity duplicated",
    "code": -33002,
    "data": {
```

```
      "email": "user-test@gmail.com",
      "reason": "user already registered"
    }
  }
}
```

Chapter 4

API: /confirm/register

This API does not follows the JSON-RPC approach because it must be invoked embedded within the body of an email and the parameters to request mus be past in the url as query parameters.

Besides, the url must be invoked with the verb GET.

Request

path: `GET /auth/confirm/register`

The url is expected to contain the following query parameters:

- `email`: the user confirming the registry
- `token`: the confirmation token generated in the call to `register`

Example

```
http://localhost:46244/auth/confirm/register?email=user-test%40gmail.com&\
token=56340b44-3014-4cca-bba1-006eef60394b
```

Response

The following http status can be returned:

- `200`: if the user account could be activated and the body has the following json

```
{
  "message": "user account user-test@gmail.com activated",
  "result": {
    "dateRegister": "2017-11-27T17:04:31.854Z",
    "email": "user-test@gmail.com"
  }
}
```

- `400`: this can be generated in case some query parameter is missing and the body has the following json

```
{
  "message": "query parameter is required",
  "parameter": "email"
}
```

- **404**: if the user account does not exist or it is already activated

```
{
  "email": "user-test@gmail.com",
  "reason": "user may not exist or it is already registered or the token is invalid",
  "token": "fc02fc19-ac9e-4dfd-a705-b7a623f8001f"
}
```

Chapter 5

API: login

The method `login` validate the user credentials and if they are valid it returns a JWS token with the authorization claims.

Request

path: `POST /auth`

method: `login`

The body must be a valid JSON-RPC request without parameters (params field empty or omitted).

The request use basic access authentication method to receive the user credentials.

This method is also protected with an API key which is expected to be provided in the http header `X-API-KEY`.

Example

```
http  -v POST http://localhost:34074/auth  \
  'Authorization: Basic dXNlci1OZXN0QGdtYWlsLmNvbTpwYXNzd29yZA==\
  'X-API-KEY: apikey'  <<< '{"jsonrpc": "2.0", "method":"login", "id":0}'
```

Response

The body of the response is a JSON-RPC object which could be a valid result in case of success or error in other case.

Success

The result field of the response contains the JWS token generated.

```
{
  "id": 0,
  "jsonrpc": "2.0",
  "result": {
    "email": "user-test@gmail.com",
```

```
    "token": \
    "eyJhbGciOiJSUzI1NiIsImtpZCI6Ikd0M1JUdXFXZEtfQ3ZscWFHUmEyOHdFdmJTRnByUHNrNm9TMWNYdG\
    l5NzAifQ.eyJhZG1pbiI6ZmFsc2UsImV4cCI6MTU0MzQzODE3NSwiaWF0IjoxNTExOTAyMTc1LCJwZXJtaX\
    NzaW9uIjp7ImdpZG1sIjp7Im1heGNwdSI6MTAsIm1heHNpemUiOjEwNzM3NDE4MjR9fSwic3ViIjoidXNlc\
    i10ZXN0QGdtYWlsLmNvbSJ9.ShyoY4___ewi_sslCsoiiVk7VZ_HFzD_HN_zNwQ7Aj0RcSD-z5hwy4BawWo\
    Vu1XV3BS2LVoGZZr_pGK6PfDEY5pmPxIh4qHtNSiZMRQt4cbjC-NRPexJagyvmz9sOqOO1fQ5lxoye-BGZP\
    -xEPPYE2wBmscGmzLegf8rGv9TJ7Q"
  }
}
```

The payload of the token contains the following claims:

- `admin`: boolean field indicating if the account has admin rights.
- `exp`: the expiration time on or after which the JWT MUST NOT be accepted for processing.
- `iat`: the time at which the token was issued.
- `permission`: the permission field of the user account.
- `sub`: the account identifier.

The following is an example of payload:

```
{
  "admin": false,
  "exp": 1543438175,
  "iat": 1511902175,
  "permission": {
    "gidml": {
      "maxcpu": 10,
      "maxsize": 1073741824
    }
  },
  "sub": "user-test@gmail.com"
}
```

Errors

The following error responses can be generated:

- **if no X-API-KEY is provided**

  ```
  {
    "error": {
      "code": -33005,
      "data": {
        "reason": "Expected X-API-KEY header"
      },
      "message": "Unauthorized"
    },
    "id": 0,
    "jsonrpc": "2.0"
  }
  ```

- **if not basic authorization is provided**

  ```
  {
    "error": {
      "code": -33005,
      "data": {
  ```

```
      "reason": "Basic authorization required"
    },
    "message": "Unauthorized"
  },
  "id": 0,
  "jsonrpc": "2.0"
}
```


- **if a wrong API key is provided**


```
{
  "error": {
    "code": -33005,
    "data": {
      "reason": "Invalid X-API-KEY header"
    },
    "message": "Unauthorized"
  },
  "id": 0,
  "jsonrpc": "2.0"
}
```


- **if wrong password**


```
{
  "error": {
    "code": -33005,
    "data": {
      "email": "user-test@gmail.com",
      "reason": "password does not match"
    },
    "message": "Unauthorized"
  },
  "id": 0,
  "jsonrpc": "2.0"
}
```


- **if user not found**


```
{
  "error": {
    "code": -33001,
    "data": {
      "email": "admin-test@gmail.comx",
      "reason": "user not found"
    },
    "message": "Entity not found"
  },
  "id": 0,
  "jsonrpc": "2.0"
}
```


- **if user account is not activated**


```
{
  "error": {
    "code": -33006,
    "data": {
```

```
      "email": "user-test@gmail.com",
      "reason": "user account need activation"
    },
    "message": "Account not activated"
  },
  "id": 0,
  "jsonrpc": "2.0"
}
```

Chapter 6

API: getPublicKeyStore

The method `getPublicKeyStore` return JWKS which is a JSON object that represents a set of JWKs. The JSON object MUST have a keys member, which is an array of JWKs. Each JWK is a JSON object that represents the public cryptographic key.

The JWKS objects is needed in order to decode the JWS token generated in the call to the method `login`.

Request

path: `POST /auth` method: `getPublicKeyStore`

The body must be a valid JSON-RPC request without parameters (params field empty or omitted).

Example

```
http  -v POST http://localhost:34074/auth  \
  <<< '{"jsonrpc": "2.0", "method":"getPublicKeyStore", "id":0}'
```

Response

The response is always a valid result JSON-RPC.

```
{
  "id": 0,
  "jsonrpc": "2.0",
  "result": {
    "keys": [
      {
        "e": "AQAB",
        "kid": "Gt3RTuqWdK_CvlqaGRa28wEvbSFprPsk6oS1cXtiy70",
        "kty": "RSA",
        "n": "jcDhk-YUAVm9W2d4Kb-0uzAiIlswsBWASWvWwqW1KW4S8EihaDoTX7FSXsH1WDfrbRgpxADF2yt0DqHwNlU9IN\
        gkVdG4ORMFwpUjOvIWuY2UYimwZvWX4UMZejSkJCnrBU3Mzq1OKX32vUYsGYe90XiFuAleHO5c4-s47w9stCs"
      }
    ]
```

```
    }
}
```

Chapter 7

API: setAdmin

The method `setAdmin` update the `admin` claim for the given user account. It can only be invoked using a valid bearer token with admin rights.

Request

path: `POST /auth`

method: `setAdmin`

The body must be a valid JSON-RPC request with the following parameters:

- `email`: the email identifying the user account.
- `admin`: a boolean value.

The method requires a bearer token in the `Authorization` header of the http request and the token must has admin rights.

Example

```
http -v POST http://localhost:45391/auth \
  'Authorization: Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6Ikd0M1JUdXFXZEtfQ3ZscWFHUmEyOHdFdmJTRnByU\
  HNrNm9TMWNYdGl5NzAifQ.eyJhZG1pbiI6dHJ1ZSwiZXhwIjoxNTQzNDE3NDQ5LCJpYXQiOjE1MTE4ODEONDksInN1Yi\
  I6ImFkbWluLXRlc3RAZ21haWwuY29tIn0.NXNLaCgyQL5u9TJMhpXnAlShRB4Ls8ZKQD9yPZnyIekq1ZV4RvhfxrhEkD\
  E8QwzyFwMFDXPuRjV--SYHYvUk2fZixc2vITsYQhjxLivGTq3PsHLREWHdqyc1GrdH53cSyjTqsve4ATwxwSxJUuDiQa\
  USWPcKlk9K5yJozcDHU18' <<< \
  '{\
      "jsonrpc": "2.0",\
      "method":"setAdmin",\
      "params": {\
        "email": "user-test@gmail.com",\
        "admin": true\
       },\
      "id":0\
   }'
```

Response

The body of the response is a JSON-RPC object which could be a valid result in case of success or error in other case.

Success

The result field of the response contains the admin clain just set and the email of the user account:

```
{
  "id": 0,
  "jsonrpc": "2.0",
  "result": {
    "admin": true,
    "email": "user-test@gmail.com"
  }
}
```

Errors

The following error responses can be generated:

- **if any of the parameters is missing**

  ```
  {
    "jsonrpc": "2.0",
    "id": "fe739f10-d44d-11e7-9a07-8d2a3b1b376f",
    "error": {
      "message": "Invalid params",
      "code": -32602,
      "data": {
        "message": "missing parameter",
        "parameter": "admin"
      }
    }
  }
  ```

- **if the `admin` parameter has the wrong type**

  ```
  {
    "jsonrpc": "2.0",
    "id": "fe73ed30-d44d-11e7-9a07-8d2a3b1b376f",
    "error": {
      "message": "Invalid params",
      "code": -32602,
      "data": {
        "message": "invalid admin paramemeter, must be Boolean",
        "parameter": "admin",
        "value": "true"
      }
    }
  }
  ```

- **if no bearer token is provided**

```
{
  "jsonrpc": "2.0",
  "id": "fe743b50-d44d-11e7-9a07-8d2a3b1b376f",
  "error": {
    "message": "Invalid JWS",
    "code": -33008,
    "data": {
      "reason": "missing bearer token"
    }
  }
}
```


- **if bearer token without admin rights is provided**


```
{
  "jsonrpc": "2.0",
  "id": "fe74d790-d44d-11e7-9a07-8d2a3b1b376f",
  "error": {
    "message": "Unauthorized",
    "code": -33005,
    "data": {
      "reason": "only admin users are allowed to modify admin status",
      "sub": "user-test@gmail.com"
    }
  }
}
```

Chapter 8

API: read profile

The method `readProfile` returns the profile object for the given user account. It can only be invoked using a valid bearer token.

Request

path: `POST /auth`

method: `readProfile`

The body must be a valid JSON-RPC request with the following parameters:

- `email`: the email identifying the user account.

The method requires a bearer token in the `Authorization` header of the http request. The sub claim of the token should match the user email or has admin rights (`admin==true`).

Example

```
http -v POST http://localhost:45391/auth \
  'Authorization: Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6Ikd0M1JUdXFXZEtfQ3ZscWFHUmEyOHdFdmJTRnByU\
  HNrNm9TMWNYdGl5NzAifQ.eyJhZG1pbiI6dHJ1ZSwiZXhwIjoxNTQzNDE3NDQ5LCJpYXQiOjE1MTE4ODE0NDksInN1Yi\
  I6ImFkbWluLXRlc3RAZ21haWwuY29tIn0.NXNLaCgyQL5u9TJMhpXnAlShRB4Ls8ZKQD9yPZnyIekq1ZV4RvhfxrhEkD\
  E8QwzyFwMFDXPuRjV--SYHYvUk2fZixc2vITsYQhjxLivGTq3PsHLREWHdqyc1GrdH53cSyjTqsve4ATwxwSxJUuDiQa\
  USWPcKlk9K5yJozcDHU18' <<< \
  '{\
      "jsonrpc": "2.0",\
      "method":"readProfile",\
      "params": {\
        "email": "user-test@gmail.com"
       },\
      "id":0\
   }'
```

Response

The body of the response is a JSON-RPC object which could be a valid result in case of success or error in other case.

Success

The result field of the response contains the email of the user account and an object profile for that user.

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "result": {
        "email": "user-test@gmail.com",
        "profile": {
            "company": "Sus labores",
            "name": "Paco",
            "otherField": "otro campo",
            "surname": "Perico"
        }
    }
}
```

Errors

The following error responses can be generated:

- **if any of the parameters is missing**

```
{
  "jsonrpc": "2.0",
  "id": "fe739f10-d44d-11e7-9a07-8d2a3b1b376f",
  "error": {
    "message": "Invalid params",
    "code": -32602,
    "data": {
      "message": "missing parameter",
      "parameter": "email"
    }
  }
}
```

- **if no bearer token is provided**

```
{
  "jsonrpc": "2.0",
  "id": "fe743b50-d44d-11e7-9a07-8d2a3b1b376f",
  "error": {
    "message": "Invalid JWS",
    "code": -33008,
    "data": {
      "reason": "missing bearer token"
    }
```

```
  }
}
```

- **if bearer token is not admin or does not match the user account**

```
{
  "jsonrpc": "2.0",
  "id": "fe74d790-d44d-11e7-9a07-8d2a3b1b376f",
  "error": {
    "message": "Unauthorized",
    "code": -33005,
    "data": {
      "reason": "not allowed to read user profile",
      "sub": "user-other@gmail.com"
    }
  }
}
```

- **if user not found**

```
{
  "error": {
    "code": -33001,
    "data": {
      "email": "admin-test@gmail.comx",
      "reason": "user not found"
    },
    "message": "Entity not found"
  },
  "id": 0,
  "jsonrpc": "2.0"
}
```

Chapter 9

API: update profile

The method `updateProfile` modify the profile object for the given user account. It can only be invoked using a valid bearer token.

Right now there is no check on the structure of the parameter profile.

Request

path: `POST /auth`

method: `updateProfile`

The body must be a valid JSON-RPC request with the following parameters:

- `email`: the email identifying the user account.
- `profile`: the new value for the profile field.

The method requires a bearer token in the `Authorization` header of the http request. The sub claim of the token should match the user email or has admin rights (`admin==true`).

Example

```
http -v POST http://localhost:45391/auth \
  'Authorization: Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6IkdOM1JUdXFXZEtfQ3ZscWFHUmEyOHdFdmJTRnByU\
  HNrNm9TMWNYdGl5NzAifQ.eyJhZG1pbiI6dHJ1ZSwiZXhwIjoxNTQzNDE3NDQ5LCJpYXQiOjE1MTE4ODE0NDksInN1Yi\
  I6ImFkbWluLXRlc3RAZ21haWwuY29tIn0.NXNLaCgyQL5u9TJMhpXnAlShRB4Ls8ZKQD9yPZnyIekq1ZV4RvhfxrhEkD\
  E8QwzyFwMFDXPuRjV--SYHYvUk2fZixc2vITsYQhjxLivGTq3PsHLREWHdqyc1GrdH53cSyjTqsve4ATwxwSxJUuDiQa\
  USWPcKlk9K5yJozcDHU18' <<< \
  '{\
      "jsonrpc": "2.0",\
      "method":"updateProfile",\
      "params": {\
        "email": "user-test@gmail.com",\
        "profile": {\
            "field": "value"
        }\
       },\
      "id":0\
   }'
```

Response

The body of the response is a JSON-RPC object which could be a valid result in case of success or error in other case.

Success

The result field of the response contains the email of the user account updated.

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "result": {
        "email": "user-test@gmail.com"
    }
}
```

Errors

The following error responses can be generated:

- **if any of the parameters is missing**

```
{
  "jsonrpc": "2.0",
  "id": "fe739f10-d44d-11e7-9a07-8d2a3b1b376f",
  "error": {
    "message": "Invalid params",
    "code": -32602,
    "data": {
      "message": "missing parameter",
      "parameter": "email"
    }
  }
}
```

- **if no bearer token is provided**

```
{
  "jsonrpc": "2.0",
  "id": "fe743b50-d44d-11e7-9a07-8d2a3b1b376f",
  "error": {
    "message": "Invalid JWS",
    "code": -33008,
    "data": {
      "reason": "missing bearer token"
    }
  }
}
```

- **if bearer token is not admin or does not match the user account**

```
{
  "jsonrpc": "2.0",
  "id": "fe74d790-d44d-11e7-9a07-8d2a3b1b376f",
```

```
  "error": {
    "message": "Unauthorized",
    "code": -33005,
    "data": {
      "reason": "not allowed to modify user",
      "sub": "user-other@gmail.com"
    }
  }
}
```

- **if user not found**

```
{
  "error": {
    "code": -33001,
    "data": {
      "email": "admin-test@gmail.comx",
      "reason": "user not found"
    },
    "message": "Entity not found"
  },
  "id": 0,
  "jsonrpc": "2.0"
}
```

Chapter 10

API: readPermission

The method `readPermission` returns the `permission` field/claim for the given user account. It can only be invoked using a valid bearer token with admin rights.

Request

path: `POST /auth`

method: `readPermission`

The body must be a valid JSON-RPC request with the following parameter:

- `email`: the email identifying the user account.

The method requires a bearer token in the `Authorization` header of the http request and the token must has admin rights.

Example

```
http -v POST http://localhost:45391/auth \
  'Authorization: Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6Ikd0M1JUdXFXZEtfQ3ZscWFHUmEyOHdFdmJTRnByU\
  HNrNm9TMWNYdGl5NzAifQ.eyJhZG1pbiI6dHJ1ZSwiZXhwIjoxNTQzNDE3NDQ5LCJpYXQiOjE1MTE4ODE0NDksInN1Yi\
  I6ImFkbWluLXRlc3RAZ21haWwuY29tIn0.NXNLaCgyQL5u9TJMhpXnAlShRB4Ls8ZKQD9yPZnyIekq1ZV4RvhfxrhEkD\
  E8QwzyFwMFDXPuRjV--SYHYvUk2fZixc2vITsYQhjxLivGTq3PsHLREWHdqyc1GrdH53cSyjTqsve4ATwxwSxJUuDiQa\
  USWPcKlk9K5yJozcDHU18' <<< \
  '{\
      "jsonrpc": "2.0",\
      "method":"readPermission",\
      "params": {\
        "email": "user-test@gmail.com"
       },\
      "id":0\
   }'
```

Response

The body of the response is a JSON-RPC object which could be a valid result in case of success or error in other case.

Success

The result field of the response contains the email and the permission field of the user account, for example:

```
{
  "id": 0,
  "jsonrpc": "2.0",
  "result": {
    "email": "user-test@gmail.com",
    "permission": {
      "gidml": {
        "maxcpu": 10,
        "maxsize": 1073741824
      }
    }
  }
}
```

Errors

The following error responses can be generated:

- **if any of the parameters is missing**

  ```
  {
    "jsonrpc": "2.0",
    "id": "fe739f10-d44d-11e7-9a07-8d2a3b1b376f",
    "error": {
      "message": "Invalid params",
      "code": -32602,
      "data": {
        "message": "missing parameter",
        "parameter": "email"
      }
    }
  }
  ```

- **if no bearer token is provided**

  ```
  {
    "jsonrpc": "2.0",
    "id": "fe743b50-d44d-11e7-9a07-8d2a3b1b376f",
    "error": {
      "message": "Invalid JWS",
      "code": -33008,
      "data": {
        "reason": "missing bearer token"
      }
  ```

```
  }
}
```

- **if bearer token without admin rights is provided**

```
{
  "jsonrpc": "2.0",
  "id": "fe74d790-d44d-11e7-9a07-8d2a3b1b376f",
  "error": {
    "message": "Unauthorized",
    "code": -33005,
    "data": {
      "reason": "not allowed to read user profile",
      "sub": "user-test@gmail.com"
    }
  }
}
```

- **if user not found**

```
{
  "error": {
    "code": -33001,
    "data": {
      "email": "admin-test@gmail.comx",
      "reason": "user not found"
    },
    "message": "Entity not found"
  },
  "id": 0,
  "jsonrpc": "2.0"
}
```

Chapter 11

API: updatePermission

The method `updatePermission` modify the `permission` field/claim for the given user account. It can only be invoked using a valid bearer token with admin rights.

Request

path: `POST /auth`

method: `updatePermission`

The body must be a valid JSON-RPC request with the following parameter:

- `email`: the email identifying the user account.
- `permission`: the new value of the permission field.

The method requires a bearer token in the `Authorization` header of the http request and the token must has admin rights.

Example

```
http -v POST http://localhost:45391/auth \
  'Authorization: Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6IkdOM1JUdXFXZEtfQ3ZscWFHUmEyOHdFdmJTRnByU\
  HNrNm9TMWNYdGl5NzAifQ.eyJhZG1pbiI6dHJ1ZSwiZXhwIjoxNTQzNDE3NDQ5LCJpYXQiOjE1MTE4ODE0NDksInN1Yi\
  I6ImFkbWluLXRlc3RAZ21haWwuY29tIn0.NXNLaCgyQL5u9TJMhpXnAlShRB4Ls8ZKQD9yPZnyIekq1ZV4RvhfxrhEkD\
  E8QwzyFwMFDXPuRjV--SYHYvUk2fZixc2vITsYQhjxLivGTq3PsHLREWHdqyc1GrdH53cSyjTqsve4ATwxwSxJUuDiQa\
  USWPcKlk9K5yJozcDHU18' <<< \
  '{\
      "jsonrpc": "2.0",\
      "method":"updatePermission",\
      "params": {\
        "email": "user-test@gmail.com",\
        "permission": {'allowX': true}\
       },\
      "id":0\
   }'
```

Response

The body of the response is a JSON-RPC object which could be a valid result in case of success or error in other case.

Success

The result field of the response contains the email of the user account updated.

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "result": {
        "email": "user-test@gmail.com"
    }
}
```

Errors

The following error responses can be generated:

- **if any of the parameters is missing**

```
{
  "jsonrpc": "2.0",
  "id": "fe739f10-d44d-11e7-9a07-8d2a3b1b376f",
  "error": {
    "message": "Invalid params",
    "code": -32602,
    "data": {
      "message": "missing parameter",
      "parameter": "email"
    }
  }
}
```

- **if no bearer token is provided**

```
{
  "jsonrpc": "2.0",
  "id": "fe743b50-d44d-11e7-9a07-8d2a3b1b376f",
  "error": {
    "message": "Invalid JWS",
    "code": -33008,
    "data": {
      "reason": "missing bearer token"
    }
  }
}
```

- **if bearer token without admin rights is provided**

```
{
  "jsonrpc": "2.0",
  "id": "fe74d790-d44d-11e7-9a07-8d2a3b1b376f",
```

```
  "error": {
    "message": "Unauthorized",
    "code": -33005,
    "data": {
      "reason": "only admin users are allowed to update permission",
      "sub": "user-test@gmail.com"
    }
  }
}
```

- **if user not found**

```
{
  "error": {
    "code": -33001,
    "data": {
      "email": "admin-test@gmail.comx",
      "reason": "user not found"
    },
    "message": "Entity not found"
  },
  "id": 0,
  "jsonrpc": "2.0"
}
```