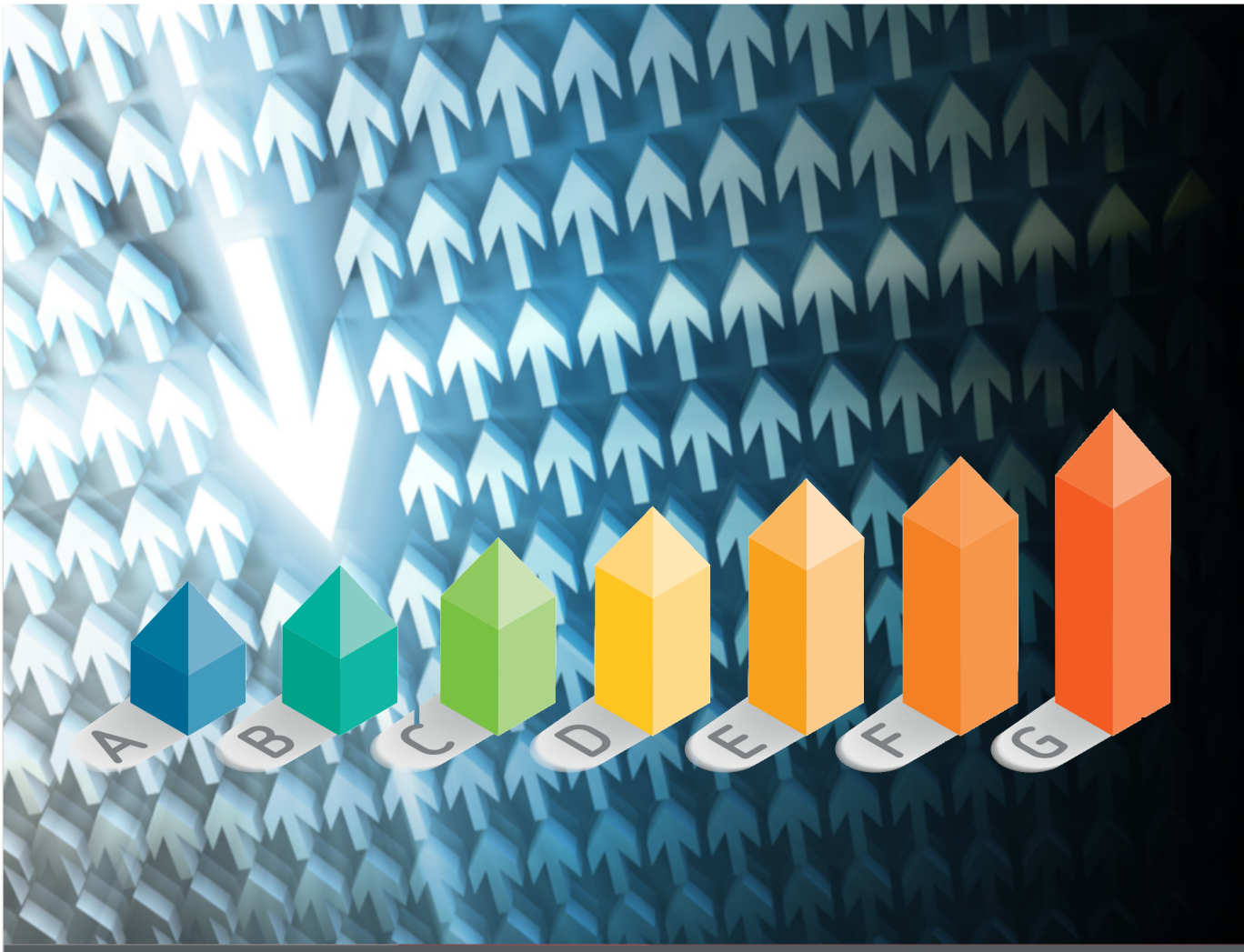
VECTOR



API Guide

Squore 19.0.17

Last updated 2020-10-26

Table of Contents

Preface

Foreword

This edition of the API Guide was released by Vector Informatik GmbH.

It is part of the user documentation of the Squore software product edited and distributed by Vector Informatik GmbH.

If you are already familiar with Squore, you can navigate this manual by looking for what has changed since the previous version. New functionality is tagged with **(new in 19.0)** throughout this manual. A summary of the new features described in this manual is available in the entry *** What's New in Squore 19.0?** of this manual's Index.

For information on how to use and configure Squore, the full suite of manuals includes:

User Manual	Target Audience
Squore Installation Checklist	New users before their first installation
Squore Installation and Administration Guide	IT personnel and Squore administrators
Squore Getting Started Guide	End users, new users wanting to discover Squore features
Squore Command Line Interface	Continuous Integration Managers
Squore Configuration Guide	Squore configuration maintainers, Quality Assurance personnel
Squore Eclipse Plugin Guide	Eclipse IDE users
Squore Reference Manual	End Users, Squore configuration maintainers
Squore API Guide	End Users, Continuous Integration Managers
Squore Software Analytics Handbook	End Users, Quality Assurance personnel

You can also use the online help from any page when using the Squore web interface by clicking **? > Help**.

Licence

Warranty

consequential damages in connection with the furnishing, performance or use of this material.

This edition of the API Guide applies to Squore 19.0.17 and to all subsequent releases and modifications until otherwise indicated in new editions.

Responsabilities

Approval of this version of the document and any further updates are the responsibility of Vector Informatik GmbH.

Contacting Vector Informatik GmbH Product Support

If the information provided in this manual is erroneous or inaccurate, or if you encounter problems during your installation, contact Vector Informatik GmbH Product Support: https://support.squoring.com/

You will need a valid customer account to submit a support request. You can create an account on the support website if you do not have one already.

For any communication:

- **support@squoring.com**
- **Vector Informatik GmbH Product Support**

  Squoring Technologies - 76, allées Jean Jaurès / 31000 Toulouse - FRANCE

Getting the Latest Version of this Manual

The version of this manual included in your Squore installation may have been updated. If you would like to check for updated user guides, consult the Vector Informatik GmbH documentation site to consult or download the latest Squore manuals at https://support.squoring.com/documentation/latest. Manuals are constantly updated and published as soon as they are available.

Chapter 1. REST Client API

API allows an easy access, through any API client, to the Squore Data.

You can find the list and documentation of the available APIs on the API Documentation page.

How to activate the API feature ?

In order to use the API Features, your Squore license must have the api.export option enabled. This option is available since Squore version 19.0. If you are not sure to have the correct license (for example, if it has not been renewed after a migration), please contact the Squore Support.

Data Security

The Squore data are protected by a token. The Token's purpose is to inform the API that the bearer of the token has been authorized to access it.

Access to data via the API is limited to user rights in Squore. You will not be able to access information that you do not have rights to.

How to retrieve a Token ?

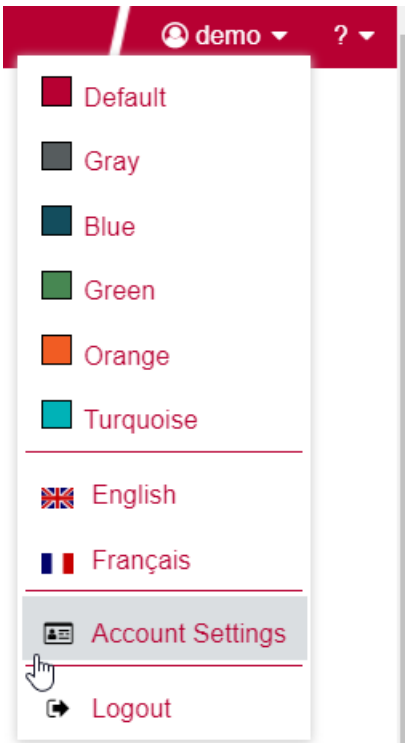
It is possible to create several tokens and name them. For example, this can be used to associate a token with an external script that would use the API to retrieve data.

There is two ways to get a token:

- By the Squore application interface
- By the Squore API

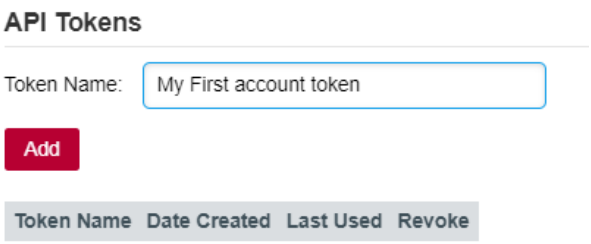
By the Squore application interface

You can create a token by clicking your user name in the menu bar and selecting the Account Settings option.



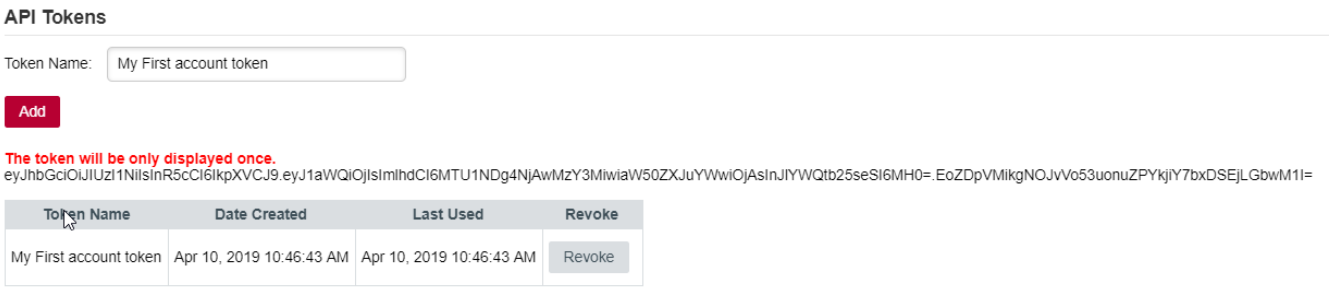
Account Settings option

In the menu, locate the "API Token" sub-menu and enter the name you want to assign to your Token.



"API Token" sub-menu

Clicking on the "Add" button will add a new permanent token in the table below.



Add a new token

You can revoke a token by clicking the "revoke" button. If so, the token will can't be used by the API anymore.

By the Squore API

You can retrieve token by API, here are examples:

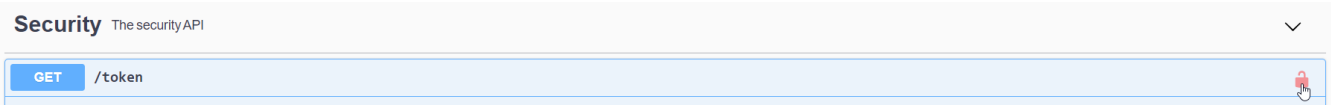
- From the API documentation (Squore swagger tool)
- From CURL external tool

- From POSTMAN external tool

Using API documentation (Squore swagger tool)

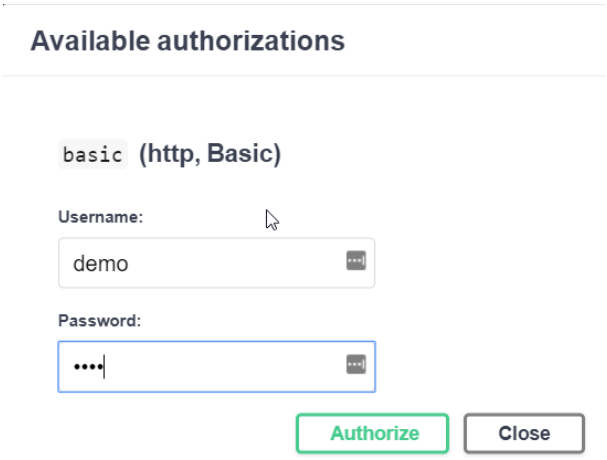
For accessing the API documentation, click **? > API Documentation** in the top-right of the web interface.

Firstly, you need to click on the padlock to authenticate yourself in swagger. Fill your login and password and click on the "Authorize" button. Then, you will be authenticate in swagger

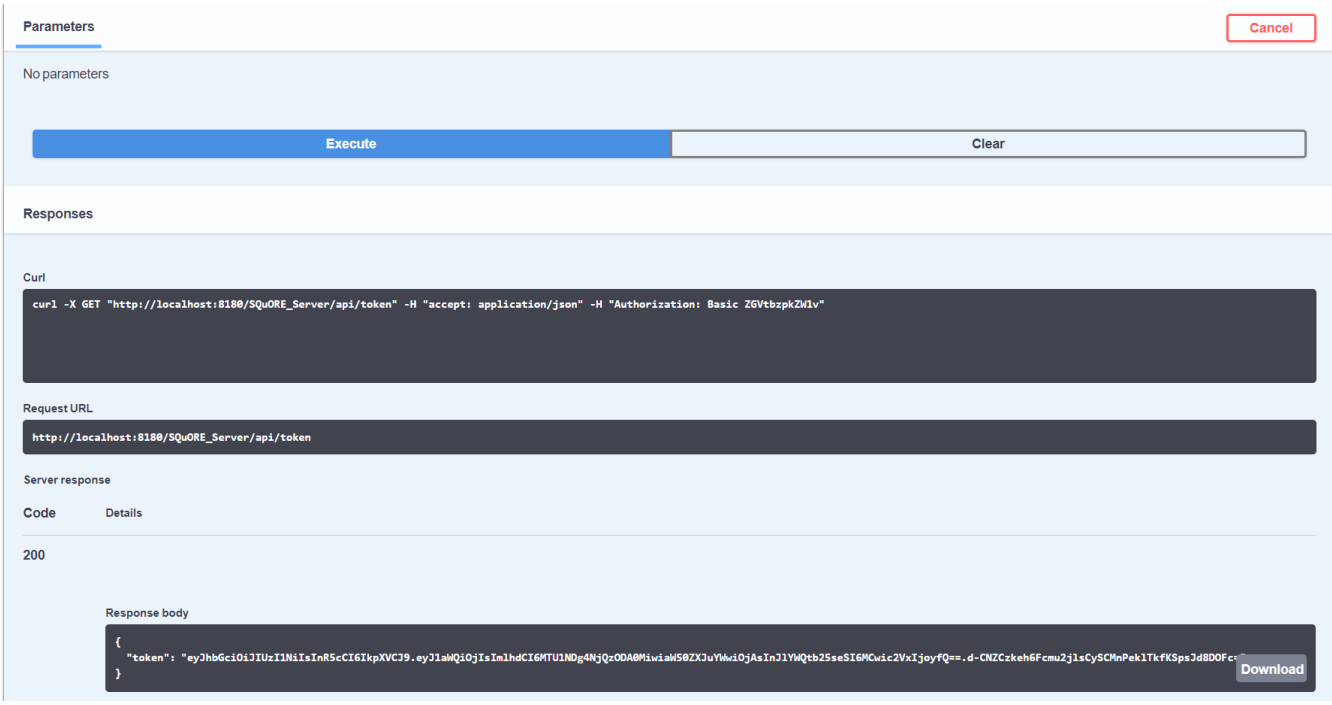


swagger authentication

In order to get a token, please find in the documentation the **GET /token** function.



For execute the API request, click on the "Try out" button, then, click on the "Execute" button which appeared in the window. In the **Server Response** section, you the token is displayed in the Response body next to the related HTTP code.



Using CURL external tool

Using the command line tool named CURL, it's necessary to respect the following syntax:

CURL example in order to add a new user

```
curl -X GET "http://localhost:8180/SQuORE_Server/api/token"     //specify the HTTP
method and the URL API
-H "accept: text/plain"         // Optional, specify the format we want (only text and
json are available).  The default value is json so this line is optional.
-su demo:demo                   // set the login and password (login:password)
```

Using POSTMAN external tool

Fill the URL API and set the HTTP method to GET. On the Authorization tab, choose the Basic Auth type and fill the login/password information

+ image::CFG_api_postman_token_login.png[]

+ If you want an other format than json, you can specified that you want text/plain format. Then click on send button to get the token into the body tab.

+ image::CFG_api_postman_token_header.png[]

Using these methods, the token is expirable (it will expire after 15 min of inactivity or 1 hour after creation).

How to call an API ?

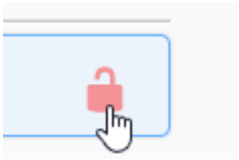
API documentation example (Swagger)

For accessing the API documentation, click ? > API Documentation in the top-right of the web interface. You will need a token to move forward. If you don't remember how to get one, please refer to How to retrieve a Token ?.

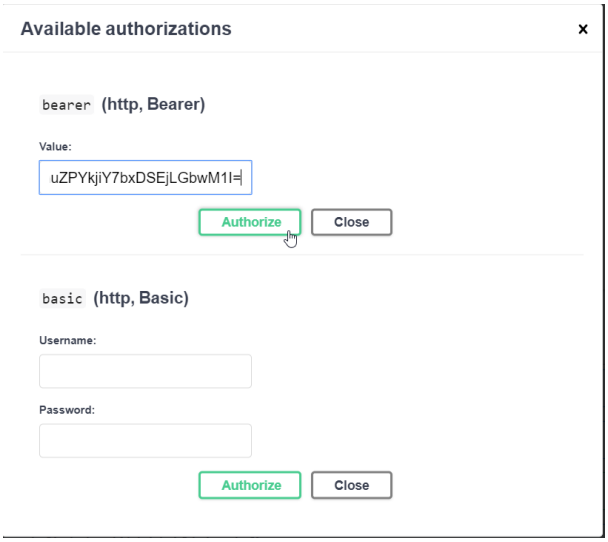
There are two ways to add your token in swagger in order to use the API:

- By clicking on the **Authorize** button

- By clicking on the padlock button

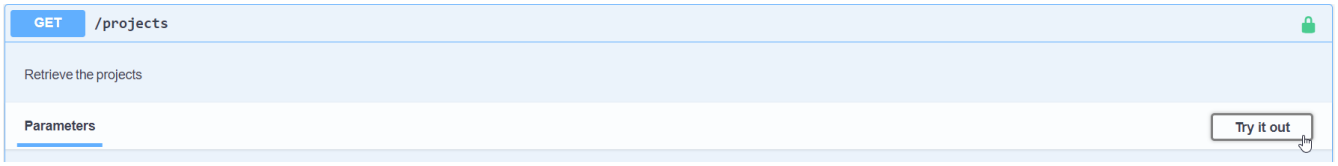


It is needed to copy the generated token in the appropriate field and click on the **Authorize** button.

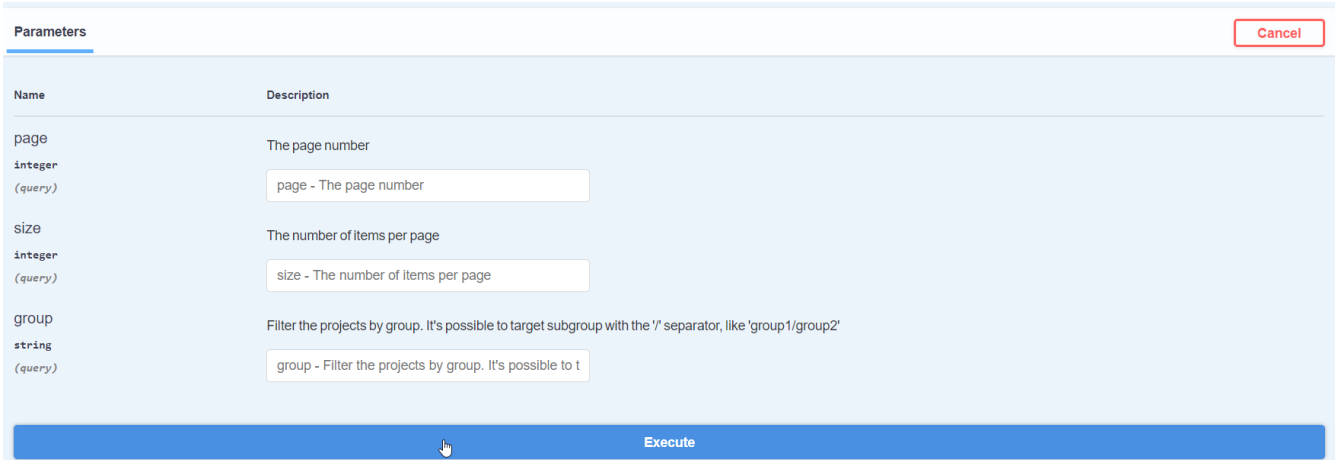


If authentication succeed, the padlocks and the "Authorize" button will turn green.

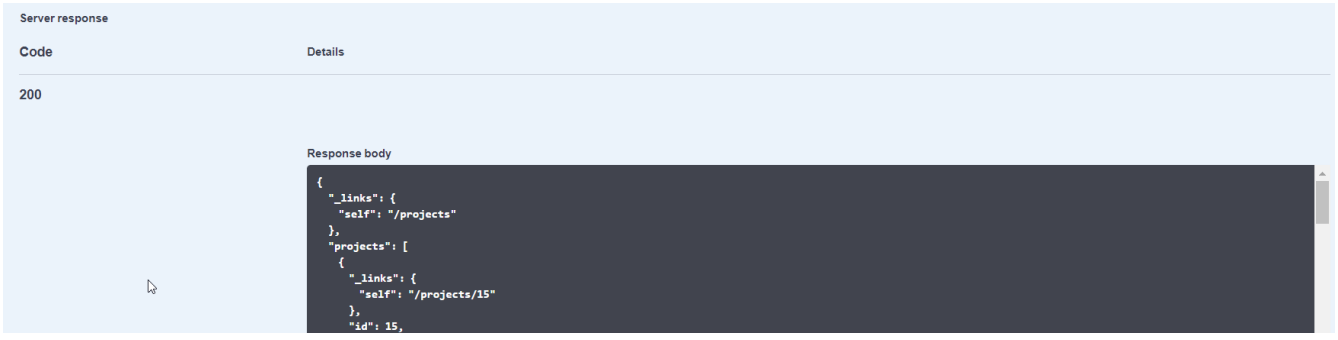
After authentication, all API are usable. Just choose an API (**Get projects** for example) and click on the **Try out** button.



Fill parameters if needed and click on the **Execute** button



Result are displayed below in the **Response body** section.



CURL example

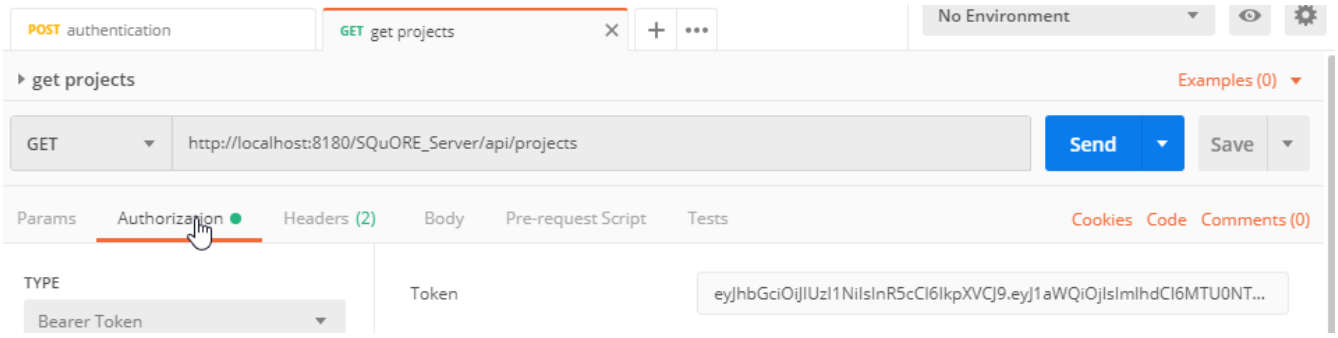
Curl example: get the project list

```
curl -X GET "http://localhost:8180/SQuORE_Server/api/projects"
-H "accept: application/json"
-H "Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOjIsImlhdCI6MTU1MjQwNTgyMzYxMiwiaW50ZXJuY
WwiOjAsInNlcSI6M30=.GimdyMOvhjABUHOE4B21bX2zi3q5SEvya257yjMgtWk="
```

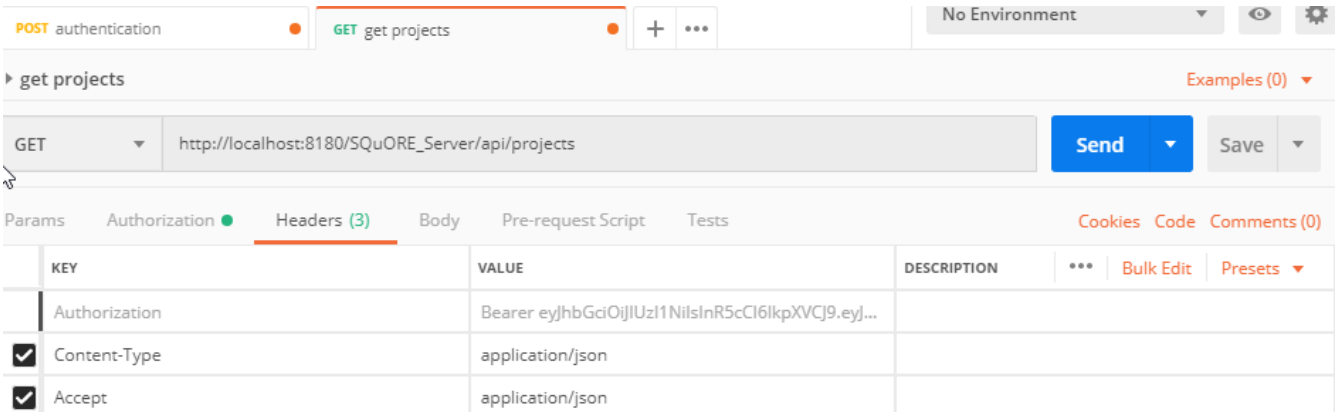
Curl example: add a new user

```
curl -X POST "http://localhost:8180/SQuORE_Server/api/users"
-H "accept: application/json"
-H "Content-Type: application/json"
-H "Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOjIsImlhdCI6MTU1MjQ2NTg0MzE2NiwiaW50ZXJuY
WwiOjAsInNlcSI6N30=.EBMsApnJ7BdIM6QYi225azgZVwb2bE7_J8TcTIEZEDQ="
-d "{\"login\":\"new-user\",\"department\":\"R&D\",\"name\":\"new-
user\",\"email\":\"demo@demo.com\",\"locale\":\"en\",\"timeZone\":\"Europe/Paris\"}"
```

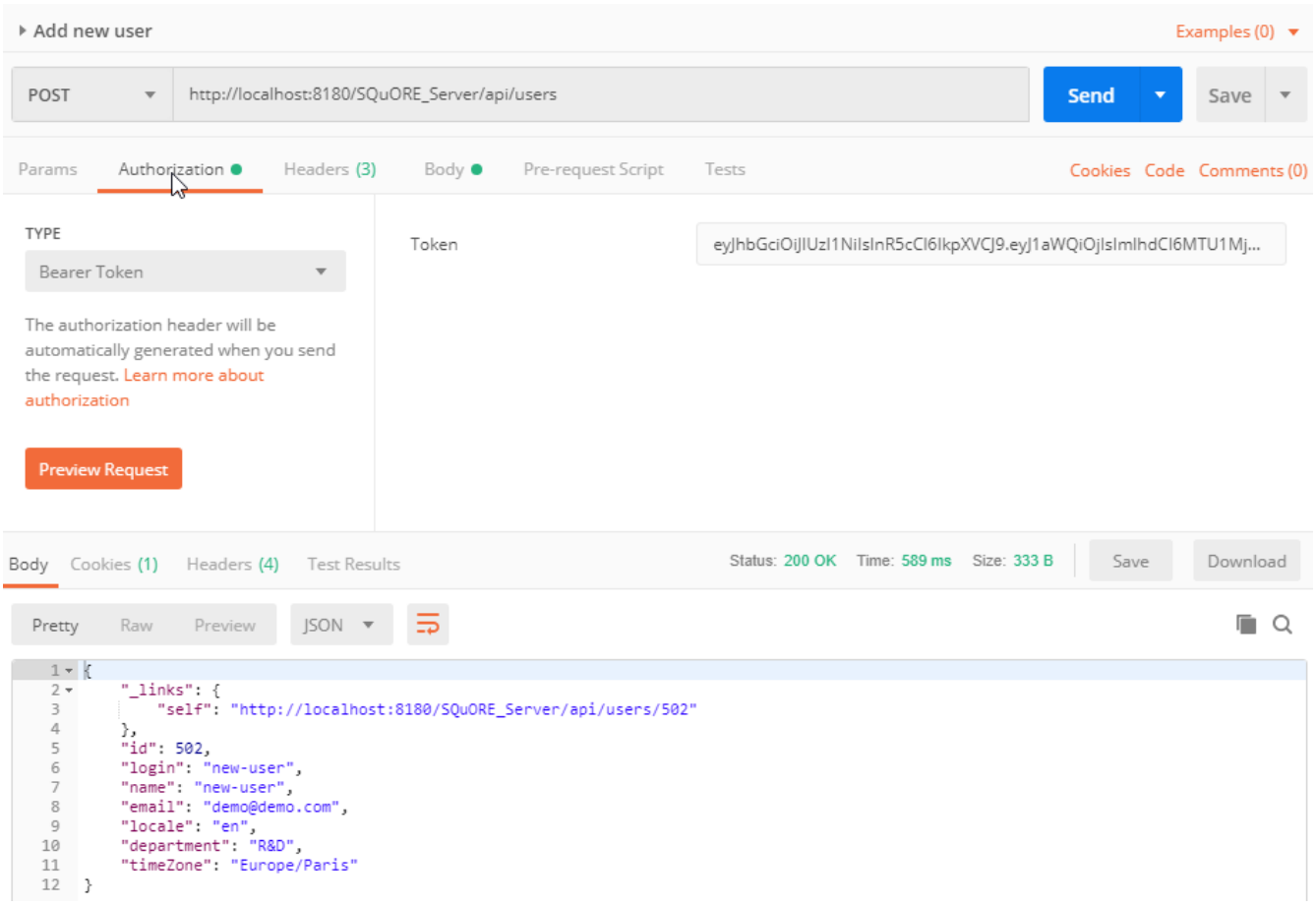
POSTMAN example



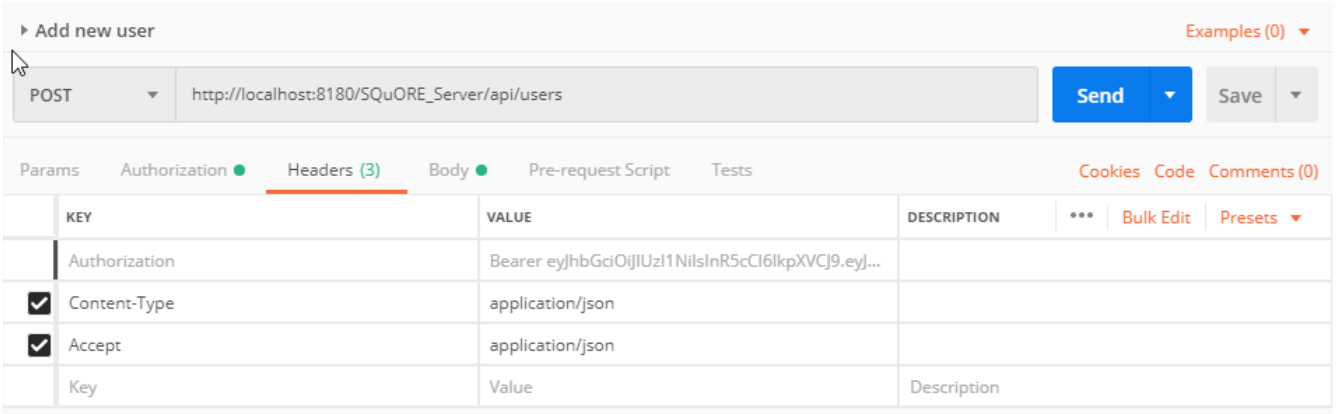
Postman example: get the project list



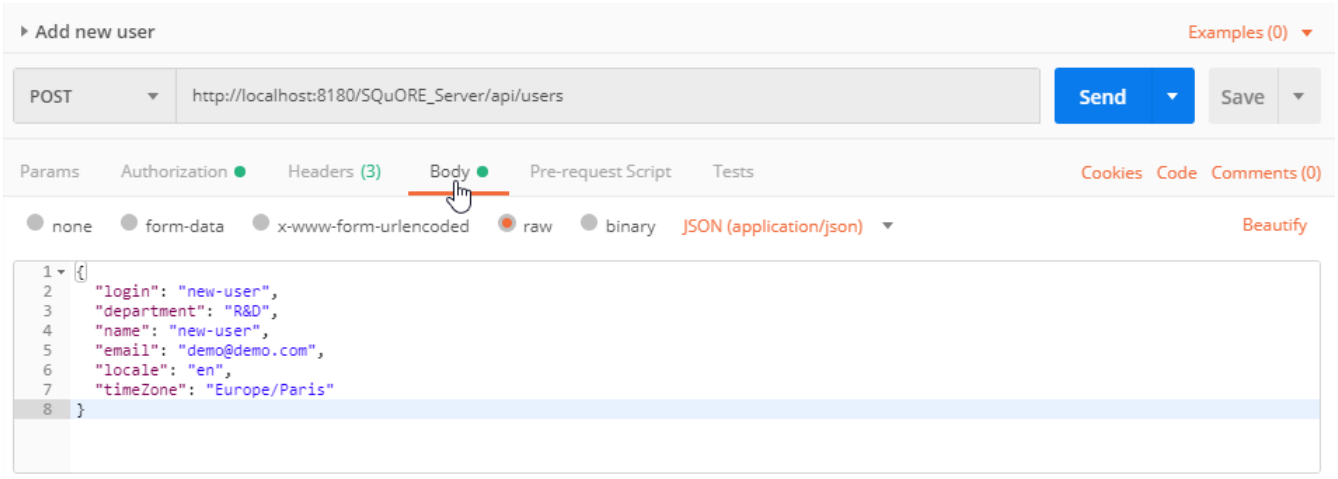
Postman header example: get the project list



Postman example: add a user



Postman header example: add a user



Postman body example: add a user

Response example

Response example of the API /projects to get the projects list


```
{
    "_links": {
        "self": "http://localhost:8180/SQuORE_Server/api/projects"
    },
    "projects": [ {
            "_links": {
                "self": "http://localhost:8180/SQuORE_Server/api/projects/9"
            },
            "id": 9,
            "name": "Mars",
            "ownerId": 2,
            "versionId": 14,
            "color": -713202,
            "status": "PENDING",
            "group": "C/",
            "creationTime": 1550487011079,
            "artefactId": 4433,
            "rating": "LEVELE",
            "modelUId": "software_analytics"
        }
    ]
}
```


APIs may change between versions of Squore.

If the API changes, the previous version will still be available in Squore. This will allow users to have time to migrate their scripts using the API. example: Squore 19.1 will be compatible with the Squore API 19.0.

API features

Links API feature

All returns items by the Squore API has an attached JSON object named "links".

The links object contains the current link (self) and the information context.

Response example of the API /projects to get the projects list


```
{
    "_links": {
        "project": "http://localhost:8180/SQuORE_Server/api/projects/9",
        "self": "http://localhost:8180/SQuORE_Server/api/artefacts/4433/findings",
        "model": "http://localhost:8180/SQuORE_Server/api/models/software_analytics",
        "version": "http://localhost:8180/SQuORE_Server/api/versions/14",
        "artefact":
"http://localhost:8180/SQuORE_Server/api/versions/14/artefacts/4433"
    },
    "findings": [{
            "_links": {
                "self": "http://localhost:8180/SQuORE_Server/api/findings/4502"
            },
            "id": 4502,
            "tool": "SQUORE",
            "definition": {
                "id": "R_NOASGCOND",
                "name": "Assignment in Boolean",
                "description": "Assignment operators shall not be used in expressions
that yield a boolean value"
            },
            "status": "OPEN",
            "locations": [{
                    "id": 2737,
                    "artefact": {
                        "_links": {
                            "self":
"http://localhost:8180/SQuORE_Server/api/artefacts/4464"
                        },
                        "id": 4464,
                        "name": "machine_plays_right()",
                        "path": "mars_engine_right.c",
                        "type": {
                            "id": "C_FUNCTION"
                        }
                    },
                    "location": "Line: 438"
                }
            ],
            "readOnly": false,
            "suspicious": false
        },...]
}
```


Pagination API feature

The pagination feature was implemented to avoid to get too much information at once. This

feature is available on all API that can retrieve a many items. To use it, we just need to pass the size and or page parameter to the query.

The curl example to get projects from the forty-first to the fiftieth

```
curl -X GET "http://localhost:8180/SQuORE_Server/api/projects?page=5&size=10"
-H "accept: application/json"
-H "Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOjIsImlhdCI6MTU1MjQwNTgyMzYxMiwiaW50ZXJuY
WwiOjAsInNlcSI6M30=.GimdyMOvhjABUHOE4B21bX2zi3q5SEvya257yjMgtWk="
```

The page and size are optional parameters.

If the size parameter was initialized and not the page, it takes by default the first page (&page=1).

On the response, several links are available to navigate between pages (current/previous/next/first/last).

The response links example to get the projects with pagination

```
{
  "_links": {
    "previous": "http://localhost:8180/SQuORE_Server/api/projects?page=4&size=10",
    "next": "http://localhost:8180/SQuORE_Server/api/projects?page=6&size=10",
    "first": "http://localhost:8180/SQuORE_Server/api/projects?page=1&size=10",
    "last": "http://localhost:8180/SQuORE_Server/api/projects?page=8&size=10",
    "self": "http://localhost:8180/SQuORE_Server/api/projects?page=5&size=10"
  },
  "projects": [{...}]
}
```

Localization API feature

The localization feature allows to localize the content of the response API.

By default, it's the language of the user that create the token. If the user doesn't have languages the english language will be chose.

But it possibles to force the language by pass the information on the header. By the same way, it possibles to change the timezone, by default is the user timezone. If the user doesn't have timezone the Europe/London timezone will be chose.

The curl example in German and Europe/Germany timezone

```
curl -X GET "http://localhost:8180/SQuORE_Server/api/projects"
-H "accept-language: en"
-H "timezone: Europe/Germany"
-H "accept: application/json"
-H "Content-Type: application/json"
-H "Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOjIsImlhdCI6MTU1MjQ2NTg0MzE2NiwiaW50ZXJuY
WwiOjAsInNlcSI6N30=.EBMsApnJ7BdIM6QYi225azgZVwb2bE7_J8TcTIEZEDQ="
```

Query Definition for API

This part describes the query definitions.

It will be useful for the Query API) to understand how to build the json body request.

First level Attributes

- `artefactTypes` **(optional, default: any)** JSON array of artefact types to filter on.
- `linkType` **(optional, default: any)** Allows following links to other artefacts. Note that only artefacts with direct links to the current artefact are displayed.
- `linkDirection` **(optional, default: OUT)** is one of **IN** or **OUT** and allows to specify the direction of the links to follow.
- `columns` **(mandatory)** JSON array of column object.
- `orders` **(optional)** JSON array of order object.
- `filters` **(optional)** JSON array of filter object.
- `resultSize` **(optional, default: all)** is the number of returning items.
- `hidePath` **(optional, default: false)** is the flag to hide the path.
- `hideRating` **(optional, default: false)** is the flag to hide the rating.

Columns Attributes

The column object is used to add a column to the list of artefacts returned.

- `type` **(mandatory)** define the type of the column. The possible values are **MEASURE**, **INDICATOR** or **INFO**.
- `id` **(mandatory)** is the ID of the measure, indicator or textual information to display.
- `displayType` **(optional, only for indicator type, default: NAME)** is the label to display in the header. The supported values are:
  - **RANK** for the indicator's rank
  - **ICON** for the indicator's rank icon
  - **NAME** for the indicator level's name
  - **MNEMONIC** for the indicator level's mnemonic
- `useBackgroundColor`**(optional, default: false)** allows you to use the indicator level as the background colour of the cell. This is only valid when using a column with `indicatorId`. You can

also control the transparency of the background colour with the attribute. Only for indicator type.

- `alpha` **(optional, default: 100)** alpha which accepts a value between 0 (transparent) and 255 (opaque). Only when useBackgroundColor is used.
- `excludingTypes` **(optional, default: none)** lists the artefact types for which the metric should not be displayed. This allows refining the types entered in the main filter above.

Orders Attributes

The order object is used to specify sorting directives.

- `type` **(mandatory)** define the measure type. The possible values are **ROOT_MEASURE** or **MEASURE**.
- `id` **(mandatory)** is the ID of the measure
- `order` **(mandatory)** is the sort order for measure. Supported values are:
  - **ASC** for ascending
  - **DESC** for descending
- `orderType` **(mandatory)** define the sorted value for the measure. Supported values are:
  - **VALUE** is the value of the measure
  - **RANK** is the rank of the measure
  - **DIFF_VALUE** is the difference between values
  - **DIFF_RANK** is the difference between ranks

Filters Attributes

The filter object is used to specify extra filtering conditions for the artefacts to return.

- `type` **(mandatory)** define the type of the column. The possible values are **measure**, **indicator** or **info**.
- `id` **(mandatory)** is the ID of the measure, indicator or textual information to return.
- `data` **(mandatory)** JSON Object that contains fields :
  - **checked** (optional,default: ?) is the values of the measure/indicator/info you want to return. (or not if invert is equals to True)
  - **bounds** (optional,only for indicators and measures types, default: [;[) is the value limits. Infinite bounds can be specified by omitting the number, e.g.: [0;[or [0;] for any null or positive number.
  - **patterns** (optional,only for info types, default: *) is the value patterns of the data you want to return. The available patterns are:
    - ***** Matches any sequence of characters in string, including a null string.
    - **?** Matches any single character in string.
- `invert` **(optional, default: false)** allows to invert the condition.

For the filter info type, you can't use the fields 'patterns' and 'checked' together. The checked field has priority.

Query definition example

```
{
    "artefactTypes": ["MODULES"],
    "linkType": "IMPLEMENTATION",
    "linkDirection": "IN",
    "hideRating": true,
    "columns": [{
            "type": "INDICATOR",
            "id": "CPXT",
            "displayType": "ICON",
            "useBackgroundColor": false
        }, {
            "type": "MEASURE",
            "id": "VG"
        }, {
            "type": "MEASURE",
            "id": "NEST"
        }, {
            "type": "MEASURE",
            "id": "NPAT"
        }, {
            "type": "MEASURE",
            "id": "STAT"
        }, {
            "type": "MEASURE",
            "id": "NOP"
        }, {
            "type": "MEASURE",
            "id": "DOPD"
        }, {
            "type": "MEASURE",
            "id": "VOCF"
        }
    ],
    "orders": [{
            "type": "MEASURE",
            "id": "CPXT",
            "order": "DESC",
            "orderType": "VALUE"
        }, {
            "type": "MEASURE",
            "id": "VG",
            "order": "ASC",
            "orderType": "VALUE"
        }
    ],
    "filters": [{
            "type": "measure",
            "id": "CPXT",
            "data": {
```

```
                "bounds": ["[0.0;["]
            }
        }
    ]
}
```

Index