QSE - API's Guide

Reference: docs.qse.group *(This guide is step-by-step guide to set up account and storage service)*

The QSE Platform API helps businesses and developers build their use-cases and cherry-pick the features and functionality they want individually.

The purpose of the Platform API is to allow businesses/white-label partners to build their own business-case on top of the patent-protected and proprietary QSE DataVault.
Endpoints are bundled into business-case related collections to help organize various work-flows.

This guide contains the step by step instructions to integrate QSE APIs to your existing systems

Table of Contents

Basic Integration Flow

As an operator integrating the DataVault system for your business purpose, the most simple integration always starts with user registration/setup.

This can be achieved in very few steps, as indicated below:

1. Register User
2. Log the new user in
3. Generate and submit 2FA Setup code
4. Submit the Master Password (Master Vault Key)

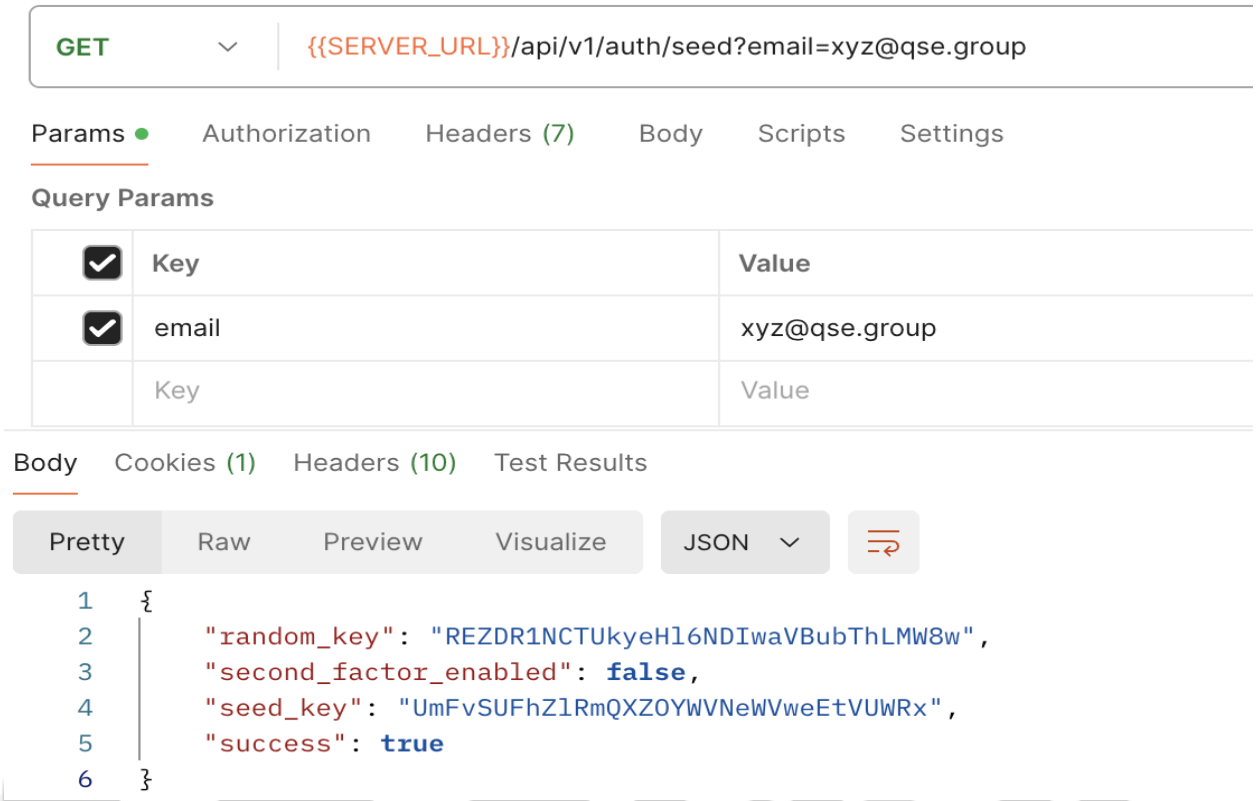
Once the above has been done (per user), further information can be submitted to the system (such as profile and personal information to activate advanced features later on).

Step 1: Get Seed

API: https://portal.qse.group/api/v1/auth/seed?email=user@qse.group

Example Usage:
Enter the email in the params section, to get the seed for that email, later you will create the password using that seed. So use the email you want to register the account with.

curl --location
'https://api.hybrid-chain.com/api/v1/auth/seed?email=user%40hybrid-chain.com'



Step 2: Create a Password

The Password must be SHA256 hashed before submission in the following way:
Example: password is “starwars”
So combine the password with seed (no space in between) and then Hash it.

Password = **‘starwars’**
Seed = **'ME00VkdxMmtxV29jY01GcnVvWlJJQ0s0XXX'**

Pseudo Code:

- var hashed_pw = SHA256(input_password+seed_key)

or

- var hashed_pw = SHA256('starwars'+'ME00VkdxMmtxV29jY01GcnVvWlJJQ0s0XXX')

A **Linux Console/Mac Terminal** example would be this:

$ **echo -n starwarsME00VkdxMmtxV29jY01GcnVvWlJJQ0s0XXX | openssl dgst -sha256**

Result:

SHA2-256(stdin)=
8f45ccbf12845c9be659d8cb1ec1973b7c291908c48969bb9f8582e855ed7da4

```
scopetech@Developers-MacBook-Pro qse-quantum-api-server % echo -n starwarsME00VkdxMmtxV29jY01GcnVvWlJJQ0s0XXX | openssl dgst -sha256
SHA2-256(stdin)= 8f45ccbf12845c9be659d8cb1ec1973b7c291908c48969bb9f8582e855ed7da4
```

So, your **hashed password** is
8f45ccbf12845c9be659d8cb1ec1973b7c291908c48969bb9f8582e855ed7da4
Use this in User Registration.

Step 3: Create a User Account (Corporate/Private)

This end-point allows a new private user to be registered/signed-up (please see the "Register a Corporate User" Endpoint for companies/corporations).

The country_code is an optional field that can be used to submit the 2-digit international country code. This may help potential KYC providers on assessing the risk-scoring system and other parameters. (Note: use the General/Countries endpoint to get a list of all valid countries).
Using the ref-code ensures that the user is correctly linked to the corresponding referring user (it can be left blank or ignored).

The Whitelabel Hash is used to assign the user to the corresponding whitelabel. if left blank, the registered user will automatically become a native QSE user.
The ref-code CANNOT be changed or amended.

API: https://portal.qse.group/api/v1/auth/registration

Example **curl** request:

Whitelabel Hash for QSE: 2f32a440e92c423baf856baff6532d53
Request your Company's whitelabel hash.
Ref_code: can be left blank (reference code)

For Private User:

```
curl --location 'https://portal.qse.group/api/v1/auth/registration' \
--data-raw '{
   "whitelabel_hash": "2f32a440e92c423baf856baff6532d53",
        "email": "new_user@company.com",
        "first_name": "John",
        "last_name": "Connor",
```

```
        "password":
"8f45ccbf12845c9be659d8cb1ec1973b7c291908c48969bb9f8582e855ed7da4",
        "accept_terms": true,
        "seed_key": "ME00VkdxMmtxV29jY01GcnVvWlJJQ0s0XXX",
        "ref_code": ""
}'
```

For Corporate User:

```
{
  "whitelabel_hash": "2f32a440e92c423baf856baff6532d53",
  "email": "abcde@qse.group",
  "company_name": "QSE",
  "password": "8f45ccbf12845c9be659d8cb1ec1973b7c291908c48969bb9f8582e855ed7da4",
  "accept_terms": true,
  "seed_key": "ME00VkdxMmtxV29jY01GcnVvWlJJQ0s0XXX",
  "ref_code": "",
  "country_code":"CA"
}
```

Response:

```
{
  "clientid": "C2PU-WXFB-Q3HK",
  "status": "Account Registration Successful",
  "success": true,
  "user_hash": null
}
```

- To test in Postman, send the request in Body/raw - JSON format

Step 4: Log a User in

As described in the "AUTH: Get Seed" Endpoint, before the login credentials can be passed to the backend, the seed must be requested using the appropriate parameters.

Use the Unique User Identifier ('email' or 'msisdn' parameter) to correctly link the user profile. Before submitting the password data in this request, it must first be SHA256-hashed as follows:

Go to Step 1: generate Seed and Random key for your user email (ignore if already generated)

Pseudo Code:

- ***var hashed_password = SHA256(password + seed);***

the HASHED_PASSWORD must then be hashed again with the random_key from the "Get Seed" Request:

- ***var randomized_password = SHA256(hashed_password + random_key);***

Now the randomized password is ready to be submitted.
For verification and as an example, the clear-text password "**starwars**" results into "**2b563e6b0e91b9795bb1cd49ba3486a0cbf1311ee1d12c4c2b4cb67067069ae3**" when using seed ("**ZDlmbFl2VkNjT01nNk1xdXhoZk1QV1dS**") and
the random key ("**WlpoUWhtMjNvY2toRXlYVlBVenBXeHVr**") correctly.

Example
scopetech@Developers-MacBook-Pro qse-quantum-api-server %
echo -n starwarsZDlmbFl2VkNjT01nNk1xdXhoZk1QV1dS | openssl dgst -sha256
SHA2-256(stdin)= 81abe8a5192e7c497adda42a3cfb378b0afc743c7f668588333157bbab977b71

NOW Combine the hashed password
81abe8a5192e7c497adda42a3cfb378b0afc743c7f668588333157bbab977b71 with random key
WlpoUWhtMjNvY2toRXlYVlBVenBXeHVr
Result :
81abe8a5192e7c497adda42a3cfb378b0afc743c7f668588333157bbab977b71WlpoUWhtMjNvY2toRXlYVlBVenBXeHVr
NOW Hash the above result p

scopetech@Developers-MacBook-Pro qse-quantum-api-server %
echo -n 81abe8a5192e7c497adda42a3cfb378b0afc743c7f668588333157bbab977b71WlpoUWhtMjNvY2toRXlYVlBVenBXeHVr | openssl dgst -sha256
SHA2-256(stdin)= **2b563e6b0e91b9795bb1cd49ba3486a0cbf1311ee1d12c4c2b4cb67067069ae3**

Final Password : **2b563e6b0e91b9795bb1cd49ba3486a0cbf1311ee1d12c4c2b4cb67067069ae3**

Use this Hashed Password for login.

API: https://portal.qse.group/api/v1/auth/login

Curl Request:

```
curl --location 'https://api.hybrid-chain.com/api/v1/auth/login' \
--data-raw '{
        "email":"user@hybrid-chain.com",
        "password":"2b563e6b0e91b9795bb1cd49ba3486a0cbf1311ee1d12c4c2b4cb67067069ae3",
        "randomkey":"WlpoUWhtMjNvY2toRXlYVlBVenBXeHVr"
}'
```

Response:

```
{
  "bearer_token": "TmdQRWtyUzZBbmRSRWw5enFqdTlxeEk4",
```

"message": "Credentials are valid, Login successful.",
"success": true
}

Use the bearer_token in the following API calls

- **Pass the bearer_token in Authorization**

Step 5: Set up 2FA Authentication

To start the 2FA Initiation process, a 2FA Initiation Key must be generated. This endpoint provides the text and image (QR Code) version as a response, which the user must import/scan in Google Authenticator (or similar).
Then the user must submit the first PIN code to fully enable and "link" the Mechanism.

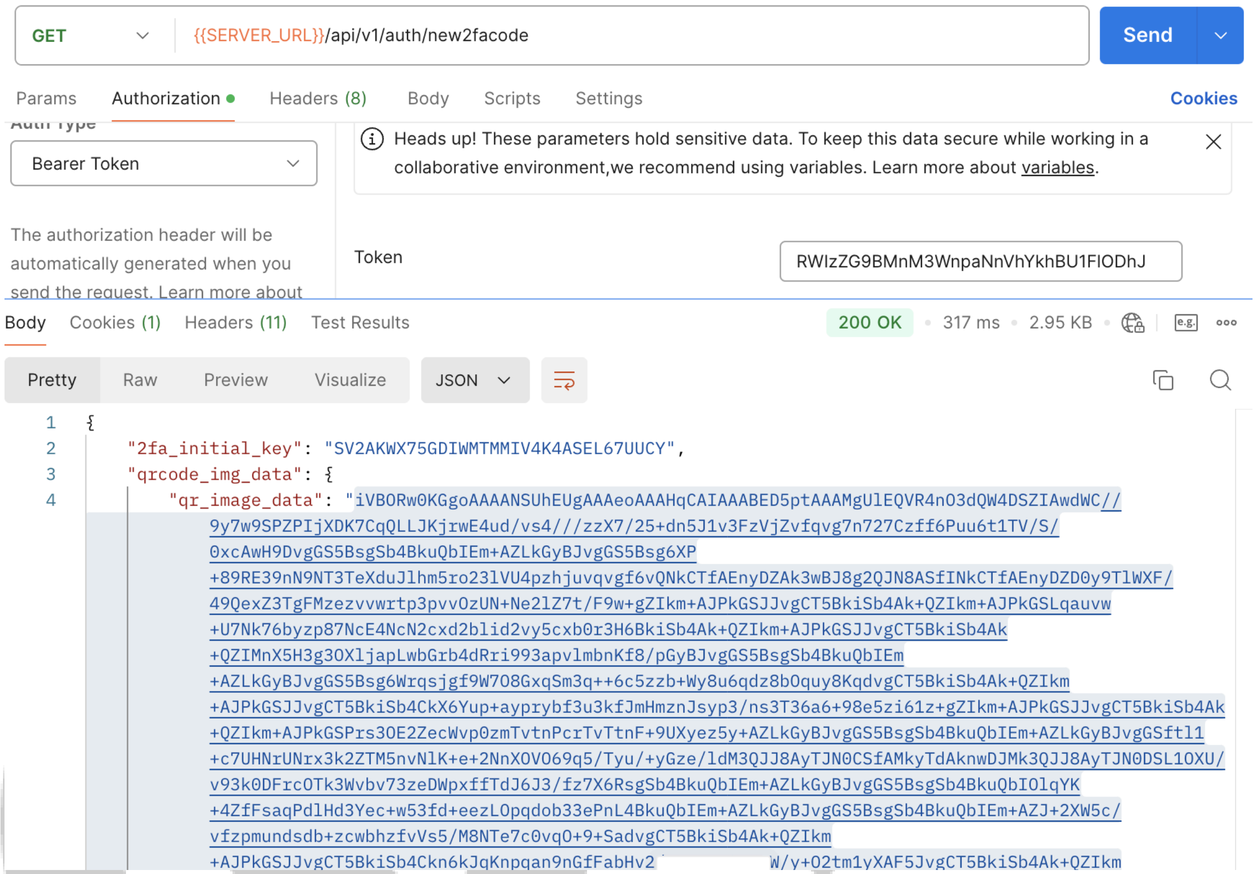
API: https://portal.qse.group/api/v1/auth/new2facode

Curl Request:
curl --location 'https://portal.qse.group/api/v1/auth/new2facode'

In Authorization
Pass the Bearer Token

Response:
JSON - contains ‘qr_image_data’ (base64)
Copy the base64 and generate image using online base64 to image converter then use your authenticator app to add the QSE account using the QR (image) generated

Example:



Steps to set up 2FA Authentication:

1. Copy the qr_image_data (Base64)
2. Go to any online Base64 to image converter (ex. https://codebeautify.org/base64-to-image-converter)
3. Paste in qr_image_data in the text box
4. Generate Image, it will show a QR Code
5. Open your Authenticator App (Google or Microsoft Authenticator)
6. Go to '+' icon in top right corner, and add work account
7. Scan the QR code
8. You will see the QSE account in your Authenticator app
9. Use the 2FA Code shown for the 2FA Authentication (for login and authentication)
10. Use the following API to submit the 2FA Code for FIRST time
    a. https://portal.qse.group/api/v1/auth/submitfirst2facode
    b. curl --location 'https://portal.qse.group/api/v1/auth/submitfirst2facode' \
       --data '{
       "first_2fa_code":123456
       }'
    c. Add bearer_token in authorization code
11. You will get a Success Message that "2FA has been enabled"

Step 6: Set up Master Password

The Master Password acts as a recovery key in the event where the 2-FA Device or Biometrics are lost or inaccessible.

The Master Password has the ability to cryptographically reset these Multi-Factor Authentication measures, so they can be re-instated.

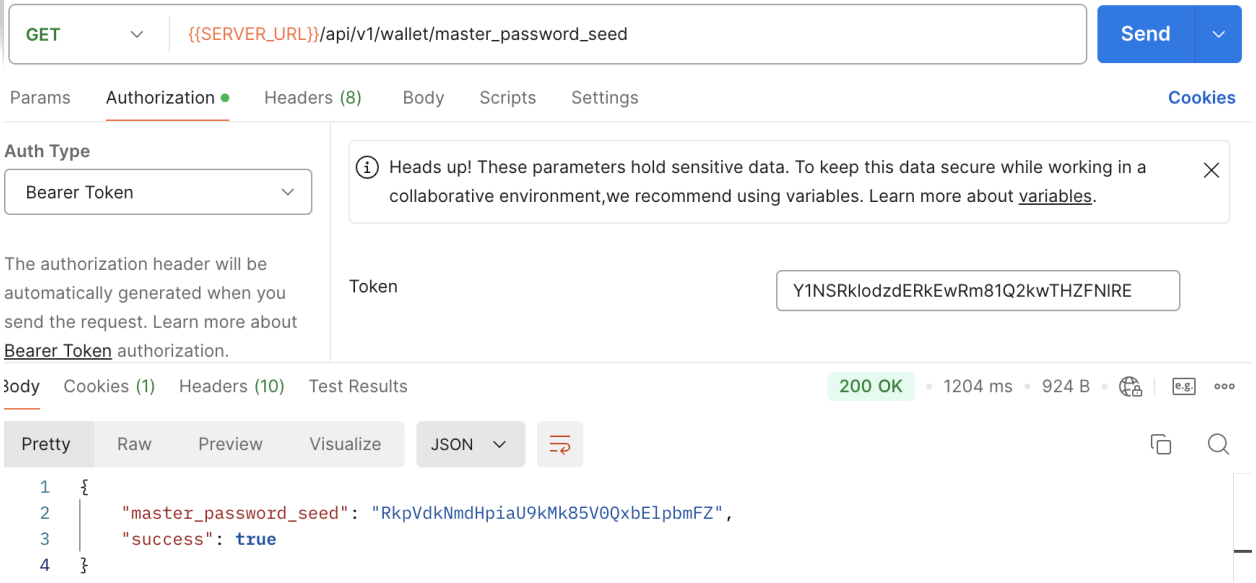
The Master Password cannot be changed once set and must be stored in a secure location to avoid any unauthorized third-parties from accessing the wallet and corresponding funds.

Submitting the Master Password requires a Master Seed to be generated.

Please use the "Get Master Seed" Endpoint to receive the Master Seed. (Follow Step A -> B)

A. **API:** https://portal.qse.group/api/v1/wallet/master_password_seed

You must be logged in and using the BEARER Token to correctly authorize the user in order to use this function.



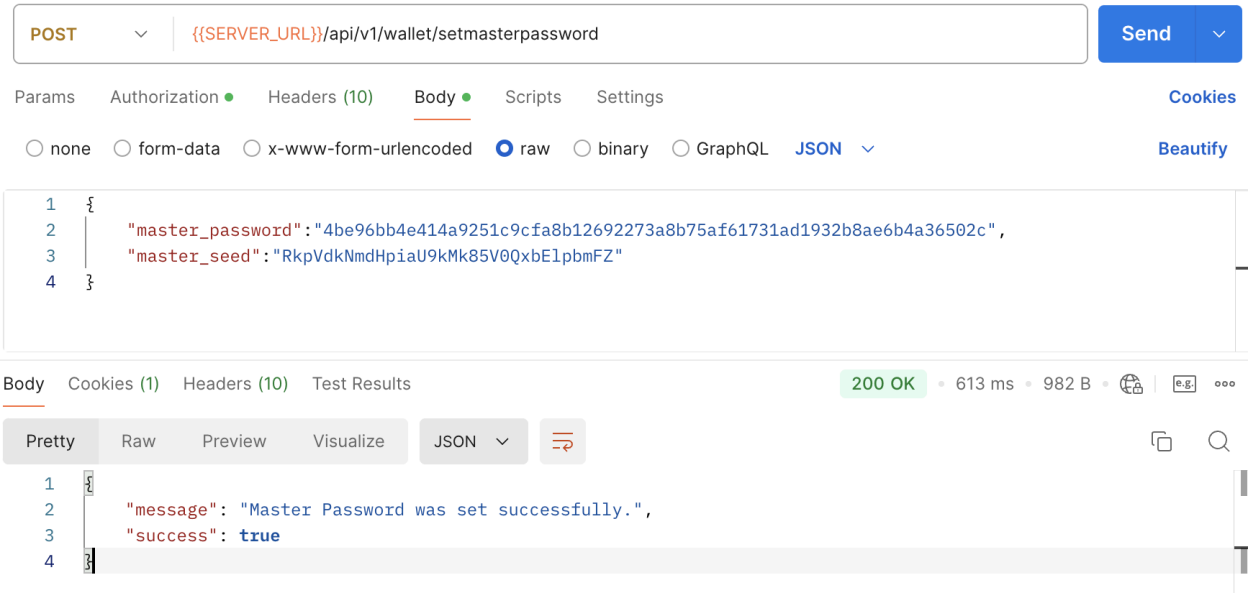
Copy the master_password_seed to generate the Hashed Password.

As explained in the Step 2: Create a Password, use master_password_seed and your_master_password to generate the Hashed Password.

B. **API**: https://portal.qse.group/api/v1/wallet/setmasterpassword

curl --location 'https://portal.qse.group/api/v1/wallet/setmasterpassword' \
--data '{

"master_password":"870f4fd3c6063f2ab058b93bc538693ed0eb867fd029acd91553f4ac0ab72e84",
"master_seed":"VzBlc2ZhaDZ6a201N2JjTXE3Qkd6SFBT"
}'



Your Account is all Set Up by Now, you can go to portal.qse.group and login to your account now.

Step 7: Upload File to Data Vault

Use this API alongside your existing data upload code, so QSE will work as a Secure Data Backup.

API: https://portal.qse.group/api/v1/quantum/storage/upload

Curl Request:

- Use Bearer_token

Use an online base64 converter to convert your data into base64 before sending.

```
Unset
curl --location
'https://portal.qse.group/api/v1/quantum/storage/upload' \
--data '[
    {
        "file_name":"Some Image2.png",
        "parent_directory": null,
        "base64_data": "ZG..."
}
}
```

Response:


```
{
  "message": [
    {
      "file_name": "Some Image2.png",
      "file_uuid": "0c8cc8cc5c3746eab4260b61cc15a0eb",
      "success": true
    }
  ],
  "success": true
}
```


Copy the file_uuid and store it in your database and for future reference.

Step 8: Get Download link for the file

API: https://portal.qse.group/api/v1/quantum/storage/download

Authorization : Bearer Token

Curl Request:

```
curl --location 'https://portal.qse.group/api/v1/quantum/storage/download' \
--data '{
    "file_uuids": [
        "c7fc265f50b14144993ade87c3ca319a",
        "bdc1755673704afc91ab96536a6519c7"
    ]
}'
```

Response:


```
[
    {
        "file_link": "http://portal.qse.group/static/downloads/Some%20Image2.png",
        "file_uuid": "a95de4a8259346f9a6b474ff1e491946",
        "success": true
    },
    {
        "file_link": "http://portal.qse.group/static/downloads/QSE%20DFD.png",
        "file_uuid": "bdc1755673704afc91ab96536a6519c7",
        "success": true
    }
]
```


Go to the link and access your file.

Step 9: Creating Folders

API: https://portal.qse.group/api/v1/quantum/storage/newfolder

Authorization: Bearer Token

```
curl --location 'https://portal.qse.group/api/v1/quantum/storage/newfolder' \
--data '{
    "folder_name": "New Folder 1",
    "working_directory": "",
    "vault_id": "b00dc220da8f480d86cdc11341372746"
}'
```

Response:


```
{
  "folder_uuid": "fb76dd17208b4f90a8e0b630e104676d",
  "success": true,
  "vault_id": "b00dc220da8f480d86cdc11341372746"
}
```


Step 10: Delete File from Data Vault

API: https://portal.qse.group/api/v1/quantum/storage/delete

Authorization: Bearer Token

```
curl --location 'https://portal.qse.group/api/v1/quantum/storage/delete' \
--data '{
    "file_uuids": [
        "c7fc265f50b14144993ade87c3ca319a"
    ]
}'
```

Response

```
{
  "success": true
}
```

Step 11: Access Web Portal Application

API: https://portal.qse.group/api/v1/auth/federate

This endpoint allows a logged in user to use their bearer token to generate a web-link that automatically logs them into the web-interface of the platform. This can be useful when needing to switch to web-only modals (KYC Systems, Payment Interfaces, etc.) while still retaining the user session context.

The returned Federation Token can be used to call the platform's login page with an optional redirect parameter:

https://portal.qse.group/login?token=your_federation_token

Note: Federation Tokens automatically expire 1 minute after creation.

Curl request:

```
curl --location 'https://portal.qse.group/api/v1/auth/federate'
Authorization: Bearer token
```

Response:

```
{
    "federation_token": "QU56QzB6bUxHZFN4MUFITEplYnY1S1Ix",
    "success": true
}
```

So, the final URL will be
https://portal.qse.group/login?token=QU56QzB6bUxHZFN4MUFITEplYnY1S1Ix

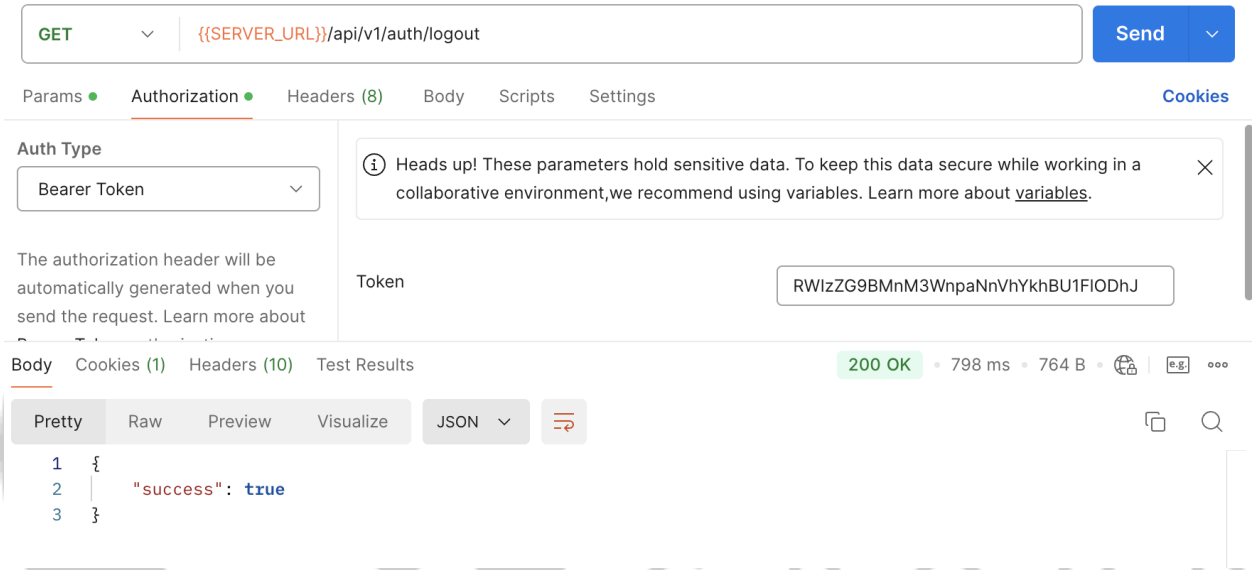
Step 12: Log out

This end-point will log the user out of any API-based sessions on all devices.

API: https://portal.qse.group/api/v1/auth/logout

Authorization: Bearer Token

Just hit the API to logout, pass your bearer token in authorization.



For More: docs.qse.group