

Televic Education

assessmentQ REST API

23 May 2023

Created by Lynn Van den Broeck



Table of contents

1	Getting started	4
1.1	About the assessmentQ REST API	4
1.2	Your connection details	4
2	Authentication and security	5
3	Using the assessmentQ REST API	6
3.1	Requests	6
3.1.1	Query string options	7
3.2	Responses	8
3.3	Data types	9
3.4	Pagination	10
4	Typical integration workflow	10
4.1	User and group management	10
4.2	Assignment management	13
4.2.1	Creating and approving an assessmentQ assignment	13
4.2.2	Linking to an assessmentQ assignment	13
4.3	Session management	15
4.3.1	Creating a session	15
4.3.2	Resuming a session	16
4.3.3	Starting the assessmentQ Player	17
4.4	Results management	20
4.4.1	Getting the answers	21
4.4.2	Additional information with respect to processing answers	24
4.4.3	Results management for assignments of type 'scorm'	25
4.4.4	Resthooks	25
5	Exploring the assessmentQ REST API using Postman or curl	27
5.1	Postman	27
5.1.1	Initial configuration	27
5.1.2	Setting up authorization	31
5.1.3	Adding a GET API call	34
5.1.4	Adding a POST API call	37
5.1.5	Additional information	41
5.2	Curl	41
5.3	Node.js	42
6	Rate limiting.....	43

7 Contact information44

1 Getting started

assessmentQ is a professional platform for online exercises, tests, assessments and exams. The assessmentQ platform provides an advanced authoring tool to structure, create and add metadata to items (20+ question types available).

With direct online publishing or a single-sign-on link to your e-learning platform, candidates can access online learning content (either at home or within a secured and controlled environment).

After the content is finished, the results can be consulted through comprehensive online reporting.

assessmentQ has also a REST API available that can be used by third party programmers to interact with assessmentQ.

1.1 About the assessmentQ REST API


The acronym "API" stands for "Application Programming Interface". An API is a defined way for a program to accomplish a task, usually by retrieving and/or modifying data. In assessmentQ's case, we provide a REST API interface for the most common actions within the application (registering a user, starting an assignment, getting back results, ...).

Developers can use the assessmentQ REST API to create applications, websites, and other projects that interact with assessmentQ. Programs talk to the assessmentQ REST API over HTTPS, the same protocol that your browser uses to visit and interact with web pages.

The assessmentQ REST API has been designed based on the OData specification. OData (Open Data Protocol) is an [ISO/IEC approved](#), [OASIS standard](#) that defines a set of best practices for building and consuming RESTful APIs. More information on OData is available through the OData website: <https://www.odata.org/>. Next to this, REST defines a set of architectural principles by which you can design Web services that focus on a system's resources, including how resource states are addressed and transferred over HTTPS by a wide range of clients written in different languages. If measured by the number of Web services that use it, REST has emerged in the last few years alone as a predominant Web service design model. In fact, REST has had such a large impact on the Web that it has mostly displaced SOAP- and WSDL-based interface design because it's a considerably simpler style to use.

It is possible to integrate with the assessmentQ demo or production environment. The demo environment is a staging environment and can be used to test and setup the integration.

1.2 Your connection details

 Important note: The information provided in this section is strictly personal and bound to your assessmentQ environment/channel. Please do not share this information.

In order to start using the assessmentQ REST API, following information will be needed for accessing your channel **CHANNEL_NAME** on the assessmentQ **DEMO|PRODUCTION** environment.

Channel base URL =

CHANNEL BASE URL

OAuth 2.0 Access Token URL =

ACCESS TOKEN URL

API Endpoint URL =

API ENDPOINT URL

Client ID =

YOUR CLIENT ID

Client Secret =

YOUR CLIENT SECRET

Scope =

assessmentQApi

More details will be provided in the next sections.

2 Authentication and security

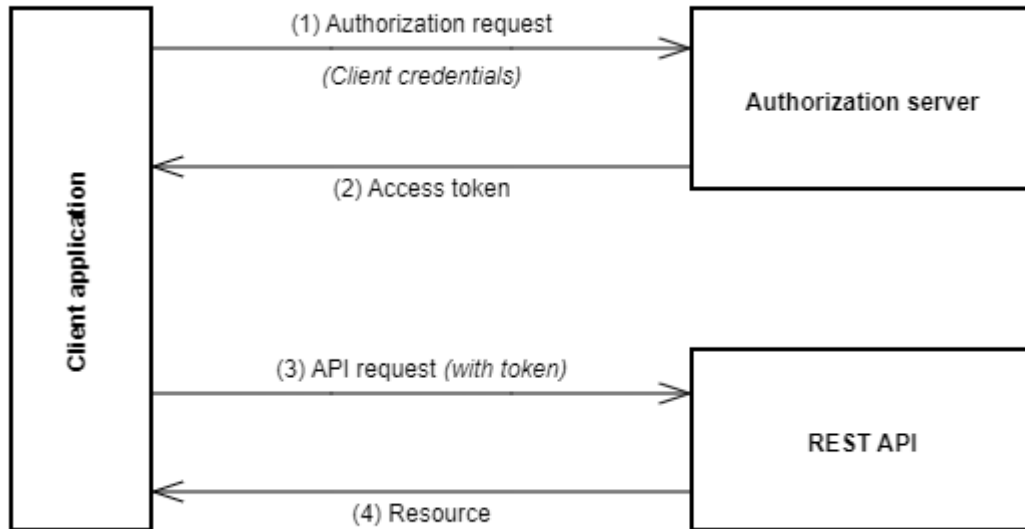
All communication is secure (HTTPS, with certificate) and all calls have to be properly authenticated using OAuth 2.0.

Therefore, prior to calling the assessmentQ REST API, an OAuth 2.0 access token needs to be obtained and added to each of the request headers. The OAuth 2.0 Access Token URL, Client ID, Client Secret, and Scope listed above must be used for requesting a new access token.

The assessmentQ REST API support the [Client Credentials Grant Flow](#) for obtaining an access token.

i The requested access token will remain valid for 10 minutes (600 seconds). After that, the token will return unusable and a new token must be requested.

Once a valid access token is obtained, the assessmentQ REST API can be used. This process is depicted (on the high level) in the picture below.



FOUT

Gliffy is zonder licentie. installeer een licentie om diagrammen in uw wiki te tekenen.

! The assessmentQ REST API is an SSL-only API (HTTPS). Therefore, all connections to the assessmentQ REST API must support the TLS 1.2 protocol and the SNI extension to TLS. Please ensure your technology stack or client library supports TLS 1.2 and SNI.

3 Using the assessmentQ REST API

A detailed overview of the available API calls can be retrieved from our automatically generated online documentation and is available by directly browsing to your <API ENDPOINT URL> url .

3.1 Requests

The assessmentQ REST API is a JSON-based API. Hence, data sent and received will be formatted in JSON. The assessmentQ REST API supports four different HTTP methods:

- GET: used to retrieve data from assessmentQ (e.g. getting a list of users using GET /v1/users)

- POST: used to add/create new data in assessmentQ (e.g. creating a user using POST /v1/users)
- PATCH: used to update existing data in assessmentQ (e.g. updating session end and start data using PATCH /v1/sessions)
- DELETE: used to delete data in assessmentQ (e.g. removing a user from a group using DELETE /v1/groups/{id}/users/{userId})

All JSON properties in the API are case insensitive.

3.1.1 Query string options

A number of OData query string options are supported. These query string options can be added to the API requests and allow controlling the amount, the order or which subset of the requested information is returned.

In the API documentation, an overview is given of the supported query string options for each of the available calls. Typically, assessmentQ REST API requests allow following query string options: *\$select*, *\$count*, *\$filter*, *\$orderby*, *\$top*, *\$skip*. See below for a number of example queries.

```
/v1/users?$select=email,username
```

Will only return the fields 'email' and 'username' of the users

```
/v1/users?$count=true
```

Will return the list of all users, including the total number of users (can be used in the case of pagination, see later). The total number of users will be returned in the @odata.count property in the resulting JSON object.

```
/v1/users?$filter=email eq 'john.doe@televic.com'
```

Returns the user(s) with e-mail address equal to 'john.doe@televic.com'.

```
/v1/users?$filter=indexof(firstName,'a') gt -1
```

Returns all the users which have the character 'a' in their first name (case insensitive).

```
/v1/users?$orderby=lastname
```

Returns all the users order by their last name (ascending).

```
/v1/users?$orderby=lastname desc
```

Returns all the users order by their last name (descending).

```
/v1/users?$top=2
```

Returns the first 2 users in the list

```
/v1/users?$skip=3
```

Skips the first 3 users in the list

Query string options can also be combined. For example, the following request will return only the fields 'email' and 'username' of all the users which have the character 'a' in their first name:

```
/v1/users?$select=email,username&$filter=indexof(firstName, 'a') gt -1
```

More information on OData query string options can also be read on the following site: <https://www.odata.org/documentation/odata-version-2-0/uri-conventions/>. Please note that the assessmentQ REST API does not support all available OData query options and functions.

3.2 Responses

Each response by the assessmentQ REST API contains an HTTP status code indicating whether the corresponding request could be fulfilled successfully or not. In general, an HTTP status code 200 indicates the request was fulfilled successfully.

In the API documentation, an overview is provided of all possible HTTP status codes for each of the available API calls.

Typical status codes are listed in the table below.

Status code	Reason
200/201/204	<p><i>Success.</i></p> <p>The original request has been processed successfully and the resulting data is returned (if any).</p> <p>HTTP status code 200 is returned in case of a successful GET and PATCH request. HTTP status code 201 is returned in case of a successful POST request. HTTP status code 204 is returned in case of a successful DELETE request.</p>
401	<p><i>Unauthorized.</i></p> <p>The API was called using an invalid OAuth 2.0 access token.</p>
403	<p><i>Forbidden.</i></p> <p>You are trying to perform an action on a channel you have no access to.</p>
404	<p><i>Not found.</i></p> <p>The requested resource cannot be found (e.g. requesting a group with an incorrect ID).</p>

Status code	Reason
422	<p><i>Unprocessable Entity.</i></p> <p>The original request could not be processed.</p> <p>In the case of an error 422, the response body will contain a field 'reasonCode' and corresponding 'message' field describing the exact error in more details. The content and associated meaning of each 'reasonCode' depends on the API call and is listed in the online documentation.</p> <p>E.g., in the case of creating a new user, following values for the 'reasonCode' and 'message' fields can be returned in case of an HTTP status code 422:</p> <ul style="list-style-type: none"> • reasonCode: 1 message: A user with emailaddress 'email' already exists. • reasonCode: 2 message: The chosen password does not meet lenght requirements. A minimum lenght of 8 characters is required. • reasonCode: 3 message: The chosen password does not meet complexity requirements.
500	<p><i>Server error.</i></p> <p>Something went wrong processing the original request on the server, an unexpected error occurred. In the case of such error, please contact the Televic Education support desk.</p> <p><u>Note:</u> in the case of an error 500, the response body will contain a 'requestId' field. Please supply this id with your message to the support desk.</p> <p>E.g.:</p> <pre> { "message": "Internal server error.", "requestId": "38e4fb43-3867-4445-a708-7c72107a448c" } </pre>

3.3 Data types

As the assessmentQ REST API is JSON-based, accepted data types are:

- string (in double quotes)
- number (integers & doubles)
- boolean (true or false)
- null
- time stamps. Time stamps use UTC time and are formatted as [ISO 8601](#) strings. For example: 2018-11-21T13:07:45.717+01:00

3.4 Pagination

When retrieving data from assessmentQ through the REST API, a maximum of 100 records/results will be returned. Pagination is used when the result contains more than 100 records/results. In this case, the resulting JSON object will contain the `@odata:nextLink` property. The value of this property contains the URL to the next page of the results.

For example, suppose there are 105 users in assessmentQ. In this case, the `/v1/users` REST API call will return the first 100 users in assessmentQ. The resulting JSON data will also contain the `@odata:nextLink` property indicating the URL to the next page of the result:

```
"@odata:nextLink": "<API ENDPOINT URL>/v1/users?$skip=100"
```

(in reality, `<API ENDPOINT URL>` will automatically be replaced by the value specified at the beginning of this document).

Hence, using the value of the `@odata:nextLink` property, calling the `<API ENDPOINT URL>/v1/users?$skip=100` function will return the last 5 users in assessmentQ. Also, the returning JSON data will now no longer contain the `@odata:nextLink` property.

Instead of using the `@odata:nextLink` property value, it is also possible to cycle through the different pages by using the `$skip` filter options in combination with the `$count` query option.

Note: Using the `$top` query option can cause the `@odata:nextLink` property not to be filled in. This happens when your `$top` is lower than the page count, which causes OData to assume you received all the data in which you were interested.

4 Typical integration workflow

In this section, we recommend a workflow for integrating access to an assessmentQ assignment (publication) into your own environment (referred to as the '3rd party environment').

Integrating with assessmentQ includes following aspects:

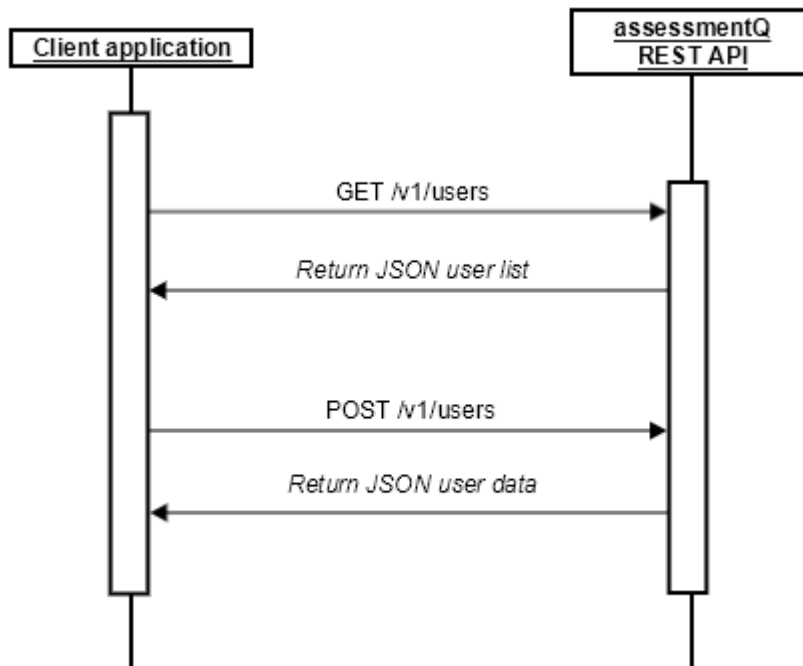
1. User and group management
2. Assignment management
3. Session management
4. Results management

Each of these aspects will be detailed in the next sections.

4.1 User and group management

Managing users and groups in assessmentQ is the responsibility of the 3rd party system. The assessmentQ REST API foresees a number of calls to facilitate querying, creating, and updating users and groups.

Creating a user in assessmentQ would typically involve following steps:



FOUT

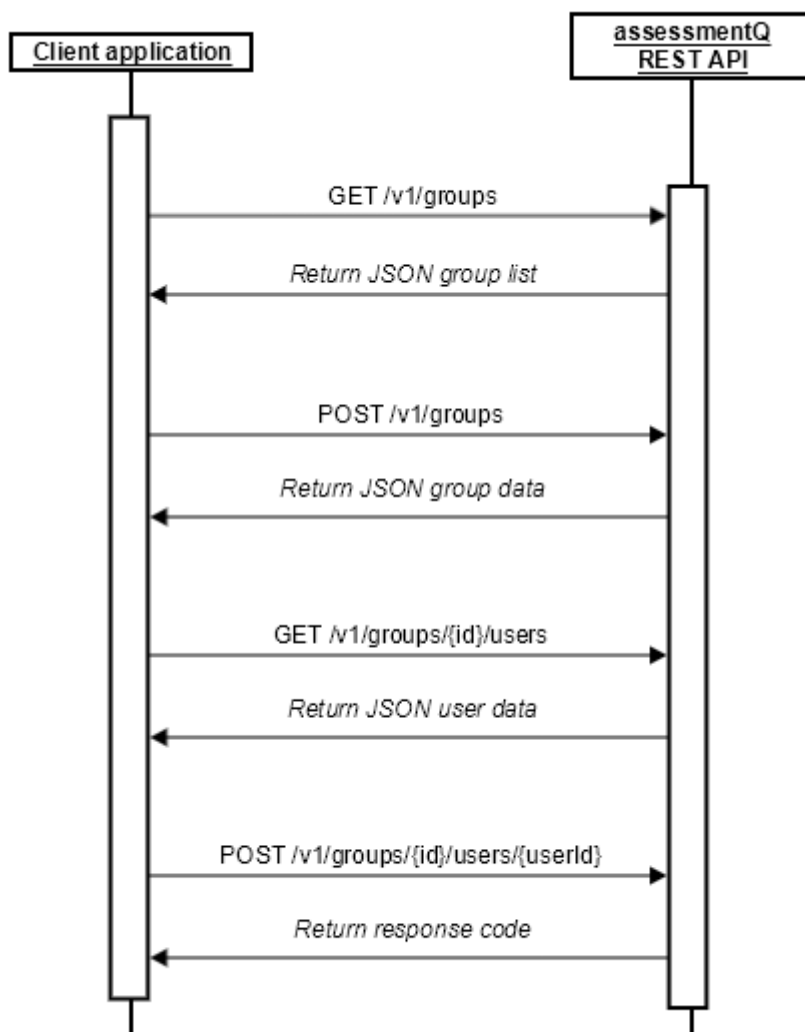
Gliffy is zonder licentie. Installeer een licentie om diagrammen in uw wiki te tekenen.

1. First, check if the particular user already exists by querying the existing users in assessmentQ. In this case, it is strongly advised to specify the \$filter query string to limit the result (e.g. /v1/users?\$filter=email eq 'john.doe@televic.com').
2. If the resulting data is an empty array, the corresponding filter did not provide any results. Hence, the user does not exist in assessmentQ.
3. In case the user does not exist, he/she can be created using the the /v1/users call (see detailed documentation)
4. When the user has been successfully created, the user details are also returned the client application.




Each user in assessmentQ has an associated id. We highly recommend storing this user id in the 3rd party system. This way, the step of verifying a user already exists in assessmentQ could be eliminated (in case this user has already an assessmentQ user id associated with it).

Once the user has been created, he/she can (optionally) be added to the necessary group(s). Groups in assessmentQ can be used to facilitate reporting and mimic the logical hierarchy in companies/environments. Use the /v1/groups calls to manage groups in assessmentQ:




 FOUT Gliffy is zonder licentie. installeer een licentie om diagrammen in uw wiki te tekenen.

1. First check whether the groups already exists in assessmentQ. It is highly recommended to specify the *\$filter* query string to limit the number of results/records (e.g `/v1/groups?$filter=name eq 'Group 01'`).
2. In case the group does not yet exist, it can be created by posting the corresponding information to `/v1/groups` (see detailed documentation).
Note: an external id can be linked to a group when creating it. This facilitates linking groups to the 3rd party system using the id of the 3rd party system.
3. Now comes the part of adding the user(s) to the group. Verify whether the user is already part of the group by querying the existing users in the group. This is achieved by executing the `/v1/groups/{id}/users` call, where `{id}` is the id of the corresponding group. The group id is returned by the `/v1/groups` call. Also in this case we recommend the usage of the *\$filter* query option to immediately search for the correct user in the group.
4. If the resulting list of users does not include the new user, he/she can be added to that group using the `/v1/groups/{id}/users/{userId}` call where `{id}` is the id of the corresponding group and `{userId}` is the id of the user to be added to the group.

 Group management is also the sole responsibility of the 3rd party system. It is the responsibility of the 3rd party system to ensure users are automatically added to/removed from groups in case something changes in the 3rd

party system. One possible solution could be to verify group membership prior to creating a new session for an assignment (see later).

4.2 Assignment management

 In our API definition you may find the term *publication* is still used widely. This is the old naming for an assignment and means the same. We chose to keep this old naming to keep supporting existing integrations.

4.2.1 Creating and approving an assessmentQ assignment

For the creation and approval of an assessmentQ assignment, please refer to the [assessmentQ Knowledge Base](#).

4.2.2 Linking to an assessmentQ assignment

In the case of a 3rd party integration, we assume a list of approved or live assignments must be maintained. Typically, the 3rd party system would offer some kind of catalog listing, among others, all available assessmentQ assignments. This catalog can then be used to offer e-learnings and e-assessments to the end-users. The available assessmentQ assignments must be known to the 3rd party system as this information is needed to create particular sessions (see next section) allowing an end-user to complete an assessmentQ assignment.

In assessmentQ, each assignment is of type 'exercise', 'assessment' or 'scorm'. By default,

- an assignment of type 'exercise' or 'scorm' can be completed in an unlimited number of times (during the time the assignment is accessible on the 3rd party platform),
- an assignment of type 'assessment' can only be completed once. However, it is the responsibility of the 3rd party system to enforce this restriction.

For more information on assignments of type 'scorm', please refer to the [assessmentQ Knowledge Base](#).

There are basically two ways of creating a link to an assessmentQ assignment from a 3rd party system. Linking to an assessmentQ assignment is done based on the assignment's ID which can either be obtained manually or programmatically. As such, it is the responsibility of the 3rd party system to save this assignment ID.

4.2.2.1 Manually linking an assignment

In order to obtain the id of an assignment, login to the assessmentQ backend and navigate to the *Assignments* module. Next, select *Approved* and *Live* in the drop-down menu to list the available approved assignments. Then select the corresponding assignment from the list and navigate to the *Properties* tab in the right panel.

Authoring / Assignments / 3.31 - All Exercise types Live Publish

Overview Items Behaviour Result

3.31 - All Exercise types

Type to find or add a tag

Details

ID	3097863
Access code	9184BE85-AFE6-4EFE-815A-15A059F488E6
Type	Exercise
Total score	792.5
Items	160
Created by	Klaas Ranson
Created on	23 Jan 2019 09:36
Modified by	Klaas Ranson
Modified on	23 Jan 2019 09:36

This assignment is live as it contains sessions that have been started.

Access

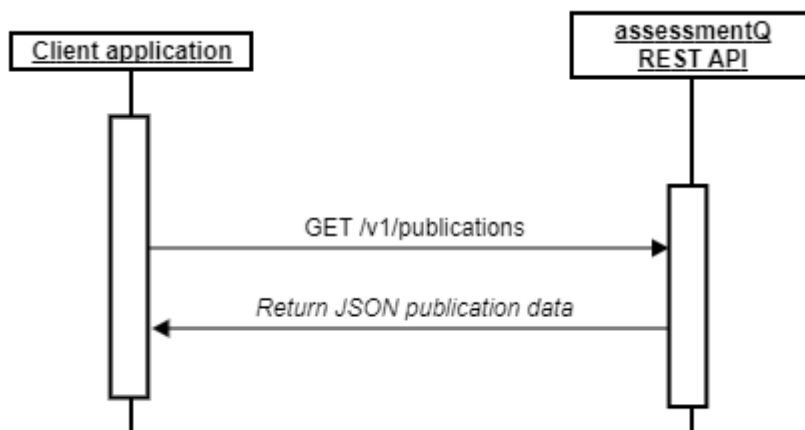
Type to find a user or group

The users you add here will be notified by e-mail that you shared the assignment with them.

Now copy the assignment id and use this to link from within the 3rd party system to the assessmentQ assignment.

4.2.2.2 Programmatically linking an assignment

It is also possible to query assessmentQ for the available assignments. This is achieved by using the /v1/publications API call.



FOUT Gliffy is zonder licentie. installeer een licentie om diagrammen in uw wiki te tekenen.

The result is a list of all assignments in assessmentQ alongside the details of each assignment:

- Id
- Name

- Description
- Type (Exercise, Assessment)
- Status (InCreation, ToBeValidated, Validated, Archived)

Typically, you would be particularly interested in the validated assignments as it is only possible to create sessions for this type of assignment. In order to directly retrieve the list of validated assignments, following query string options can be provided: `/v1/publications?filter=status eq 'Validated'`.

⚠ In V2 of our API, the term "Validated" has been split up into "Approved" and "Live" to distinguish between assignments that are ready to be used but not yet used, and ready to be used and already have started sessions.

4.3 Session management

With session management, we indicate the process of linking a particular end-user to a specific assignment he/she needs to complete. As such, a session is related to exactly one end-user and one assignment. Once an assignment is linked from within the 3rd party system (cfr. previous section), an end-user can access it.

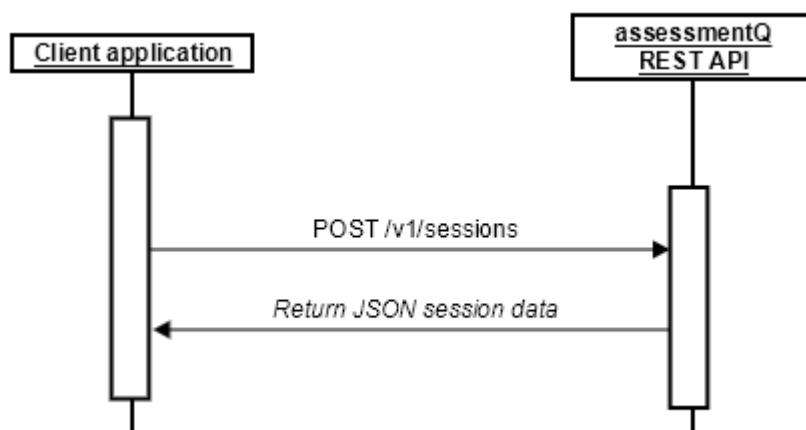
In this section, we assume the particular end-user has already been created in assessmentQ.

Once a session has been created, the access code can be entered manually by the end-user on the assessmentQ accesscode page. In case of a close integration with a 3rd party system, the assessmentQ player can be started automatically to present the assignment to that particular end-user. This way, the end-user will also automatically be authenticated so he/she can immediately start with the assignment.

This is detailed in the next subsections.

4.3.1 Creating a session

As already explained, a session is attributed to exactly one end-user and one assessmentQ assignment. Assuming the end-user is already created in assessmentQ and the correct assessmentQ assignment has been linked in the 3rd party system, a session can be created as follows.



FOUT Gliffy is zonder licentie. installeer een licentie om diagrammen in uw wiki te tekenen.

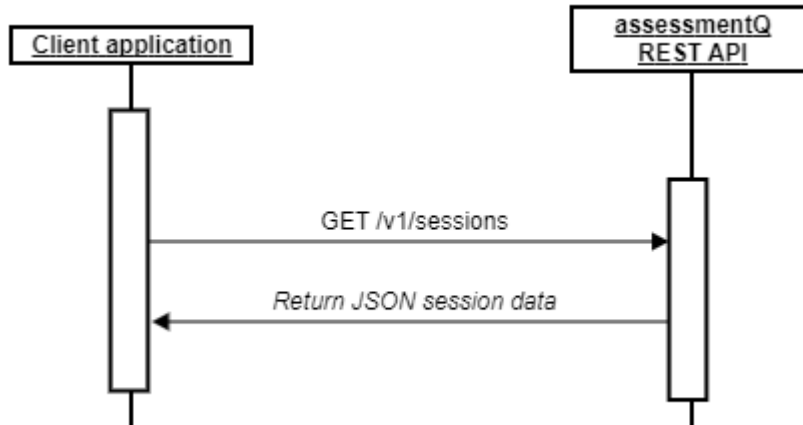
If the call completes successfully (HTTP Status Code 201), the returned JSON session data will contain the `'accesscode'` property. The value of this property is the session accesscode needed to start the assignment for that particular user (see next subsection).

For assignments of type 'assessment', an error will be returned in case no new session can be created. Ideally, this error should be caught and a proper error message should be displayed to the end-user. For example:

"You can take this test only once. Please contact your responsible if you could not complete the test in your first attempt."

4.3.2 Resuming a session

Instead of starting a new session, it is also possible to resume a session that is not yet finished. To receive a list of all sessions, the /v1/sessions API call can be used.




Example return value:

```
[{
  "id": 11858272,
  "accessCode": "123-ABC-456",
  "status": "Incomplete",
  "score": 0.0,
  "scoreMax": 38.0,
  "passed": false,
  "duration": 2,
  "interfaceLanguage": "Dutch",
  "startedOn": "2020-03-13T14:39:14.537+01:00",
  "finishedOn": null,
  "accessStart": null,
  "accessStop": null,
  "endTime": null,
  "user": {
    "id": 446319,
    "firstName": "John",
    "lastName": "Doe",
    "email": "j.doe@televic.com",
    "username": "johndoe",
    "interfaceLanguage": "English",
    "birthdate": null
  },
},
{
  "publication": {
    "id": 3721951,
    "name": "Sample publication",
    "description": null,
    "type": "Exercise",
    "status": "Validated"
  }
}
```



```
}  
}]
```

Session status is either 'Complete' or 'Incomplete' to indicate whether the session has been finished or not. In order to identify whether the session has been started or not, look at the value of 'startedOn'. If this value is *null*, the session has not been started yet.

 Please note that this call returns all sessions in the database, for each assignment and for each user. It is strongly advised to use different OData query string options to limit the result as much as possible.

For example, to return the most recent unfinished session for a particular user and assignment, following call can be used:

```
/v1/sessions?$filter=publication/id eq <publicationid> and user/id eq <userid> and status eq 'Incomplete'&$orderby=startedOn desc&$top=1
```

Where *<publicationid>* and *<userid>* need to be replaced by the assignment id and user id, respectively.

If the call above returns an empty list, no unfinished session is available and, hence, a new session needs to be created (using the POST */v1/sessions* call as detailed in the previous section).


4.3.3 Starting the assessmentQ Player

Once a session access code has been created, the assignment can be started either by entering the accesscode manually on the assessmentQ accesscode page or by automatically starting the assessmentQ player in case of a close integration with a 3rd party system.

Manually entering the accesscode

The URL of the assessmentQ accesscode page is different for the demo and production environment:

- assessmentQ DEMO environment - accesscode page URL = https://<channel_subdomain>.demo.assessmentq.com/accesscode
- assessmentQ PRODUCTION environment - accesscode page URL = https://<channel_subdomain>.assessmentq.com/accesscode

 Whether you need to use the demo or production environment is specified in Section 1.2 of this document. Please ensure you are connecting with the right environment.

When launching the assessmentQ accesscode page, the end-user will be prompted to enter his/her session accesscode. After submitting a valid session access code, the user will automatically be redirected to the corresponding assignment.

Automatically starting the assessmentQ player

By providing the access code as query string parameter to the assessmentQ player, the user will automatically be redirected to the correct assignment. Furthermore, as a session access code is strictly linked to one particular user and one assignment, the end-user will be automatically authenticated in assessmentQ.

The base URL for launching the assessmentQ player is different for the demo and production environment:

- assessmentQ DEMO environment - accesscode page URL = https://<channel_subdomain>.demo.assessmentq.com
- assessmentQ PRODUCTION environment - accesscode page URL = https://<channel_subdomain>.assessmentq.com

i Whether you need to use the demo or production environment is specified in Section 1.2 of this document. Please ensure you are connecting with the right environment.

To start the assessmentQ player with a specific session access code, supply the following query string parameters:.

```
sessionAccessCode=<SessionAccessCode>
```

For example, for the assessmentQ demo environment, the complete URL will then look as follows.

```
https://<channel_subdomain>.assessmentq.com?sessionAccessCode=123-456-789
```

Player Events

The assessmentQ player can communicate events to its parent, the top parent and the owner of the player-window. This is done by the `Window.postMessage` function (<https://developer.mozilla.org/en-US/docs/Web/API/Window/postMessage>). You can listen to these events using `window.addEventListener("message", callback)`, where `callback` is your own implementation. Following events are supported by the assessmentQ player:

- `playerstarted`: indicates the assessmentQ player has been started/launched.
- `playerfinished`: indicates the assessmentQ player has been closed.
- `sessionfinished`: indicates that the user clicked the 'finish' button in the assessmentQ player. This indicates that the session for the user has been terminated.
Note that assignments of type 'scorm' have permanent sessions. As a result, 'sessionfinished' is not supported for assignments of type 'scorm'.

The following HTML page showcases a small code example on how to integrate the assessmentQ player inside your own environment: [assessmentQPlayerIntegration.html](#) (do not forget to replace <CHANNEL BASE URL> by the information provided at the beginning of this document).

assessmentQ player integration HTML example

```
1 <html>
2   <head>
3     <meta charset="utf-8"/>
4     <title>assessmentQ Player integration example</title>
5   </head>
6
7   <body onload="initAndStartPlayer()">
8
9     <h1>assessmentQ Player integration example</h1>
10
11    <p>To start the player with a sessionAccessCode navigate to the following url: [baseUrl]?
12    sessionAccessCode=[SessionAccessCode].</p>
13
14    <p>assessmentQ DEMO environment baseUrl: https://<channel\_subdomain>.demo.assessmentq.com<br/>
15    >assessmentQ LIVE environment baseUrl https://<channel\_subdomain>.assessmentq.com</p>
16
17    <p>The assessmentQ player can communicate events to its parent, the top parent and the owner of the
18    player-window. This is done by the Window.postMessage function (https://developer.mozilla.org/en-US/docs/Web/API/Window/postMessage). You
19    can listen to these events using window.addEventListener("message", callback), where callback is your own
20    implementation. Following events are supported by the assessmentQ player:

- playerstarted: indicates the assessmentQ player has been started/launched.
- playerfinished: indicates the assessmentQ player has been closed.
- sessionfinished: indicates that the user clicked the 'finish' button in the assessmentQ player. This indicates that the session for the user has been terminated.

```

```

21     </p>
22
23     <p>In this example, the player will start in a popup (separate tab). After finishing the session, an
24     alert will be shown on this page that the session has been ended.</p>
25
26     <p><center><label for="session">Session Access Code</label> <input name="session" id="session" type="t
27     ext"/></center></p>
28
29     <p><center><button onclick="startPlayer();">Launch assessmentQ Player</button></center></p>
30
31     <script>
32     function startPlayer(){
33         var accessCode = document.getElementById("session").value;
34         if(accessCode === "")
35         {
36             alert("No valid accessCode");
37         } else
38         {
39             window.open('https://<channel_subdomain>.demo.assessmentq.com?
40             sessionAccessCode='+accessCode);
41         }
42     }
43
44     function initAndStartPlayer(){
45         window.addEventListener('message', callback);
46     }
47
48     function callback(e){
49         if (e.data === "playerstarted" || e.message === "playerstarted") {
50             document.getElementById("logmessages").innerHTML += "<br/>INFO: assessmentQ player
51             launched";
52         }
53         if (e.data === "playerfinished" || e.message === "playerfinished") {
54             document.getElementById("logmessages").innerHTML += "<br/>INFO: assessmentQ player
55             closed";
56         }
57         if (e.data === "sessionfinished" || e.message === "sessionfinished") {
58             document.getElementById("logmessages").innerHTML += "<br/>INFO: assessmentQ player session
59             finished";
60         }
61     }
62 }
63 </script>
64
65 <p>
66 <div style="border-style: solid; background-color:#dedede;">
67 <u>Events captured from the assessmentQ player:</u>
68 <div id="logmessages">
69 </div>
70 </p>
71
72 </body>
73 </html>

```

Review the session in the assessmentQ player

To review (read-only mode) a session in the assessmentQ player with a specific session access code, supply the following query string parameters:.

```
sessionAccessCode=<SessionAccessCode>&review=true
```

For example, for the assessmentQ demo environment, the complete URL will then look as follows.

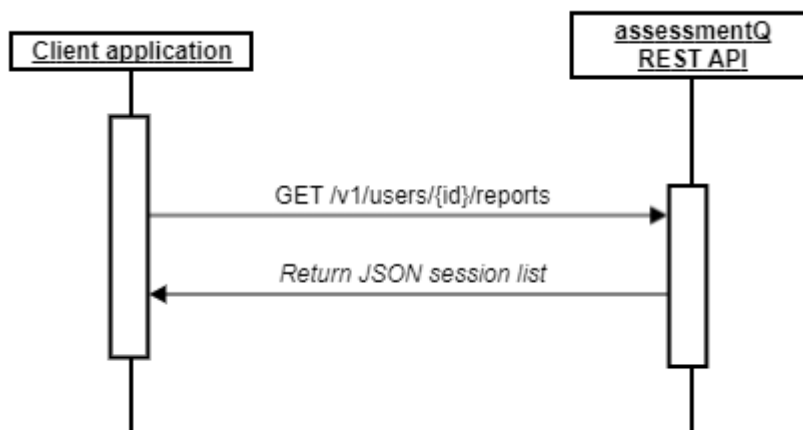
https://<channel_subdomain>.assessmentq.com?sessionAccessCode=123-456-789&review=true

i The *&review=true* parameter can even be skipped if the session is already finished. If a finished session is re-opened, the session will automatically be in review (read-only) modus.

4.4 Results management

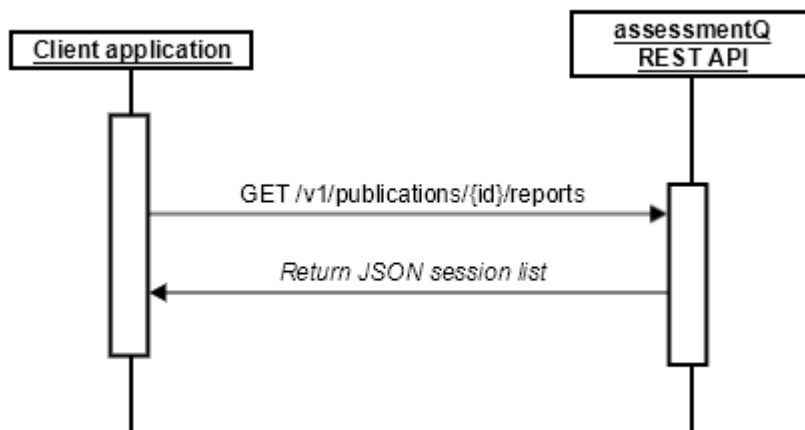
Currently, the assessmentQ REST API supports querying the results for a particular user, assignment or particular group of users using the calls */v1/users/{id}/reports*, */v1/publications/{id}/reports* and */v1/groups/{id}/reports*, respectively (where *{id}* is replaced by the id of the associated user, assignment or group).

i Please note that results can only be obtained from approved or live assignments.

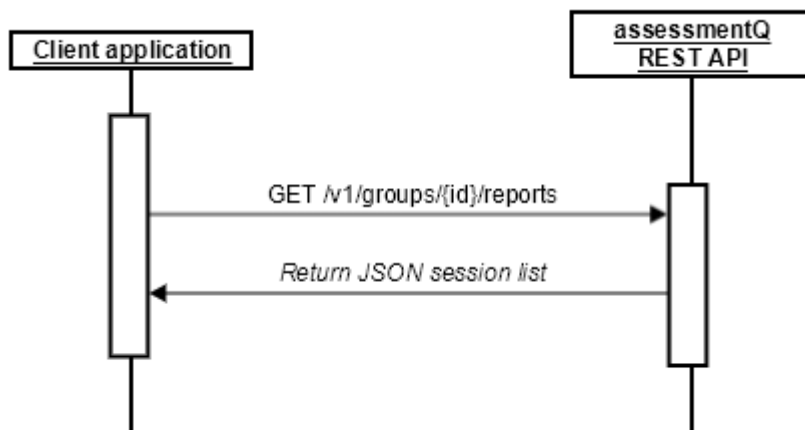


FOUT

Gliffy is zonder licentie. installeer een licentie om diagrammen in uw wiki te tekenen.



 FOUT | Gliffy is zonder licentie. installeer een licentie om diagrammen in uw wiki te tekenen.



 FOUT | Gliffy is zonder licentie. installeer een licentie om diagrammen in uw wiki te tekenen.

In both cases, the result will contain a list of all sessions (cfr. Session management) for that assignment or group of users. Each session will report the name of the assignment and identification of the user this session uniquely belongs to. Also, the obtained score will be reported both on the level of the assignment as on the level of each component inside the assignment.

In order to get the report for a specific session, use the OData *\$filter* query string. For example:

```

/v1/publications/417248/reports?$filter=accessCode eq 'J77-MVN-E2Q'
  
```

You can use the OData *\$expand* to retrieve a more detailed report, with scores for each individual item in the session (grouped by component). In order to do so, use *\$expand=items*, for example: */v1/publications/{id}/reports?\$expand=items*.

4.4.1 Getting the answers

Next to reporting the obtained scores, the assessmentQ REST API also supports obtaining the individual answers and solutions to each of the questions/items in the assignments/components. The answers can be obtained using the */v1/publications/{id}/reports?\$expand=answers* and */v1/groups/{id}/reports?\$expand=answers* calls, respectively (where *{id}* is replaced by the id of the associated assignment or group). In this case, both the scores, answers, and correct solutions will be returned per item, grouped per component.

For example:

```
[{
  "id": 4562947,
  "accessCode": "ACS-379-XCY",
  "status": "Incomplete",
  "score": 0.0,
  "scoreMax": 11.0,
  "passed": false,
  "duration": 44,
  "interfaceLanguage": "English",
  "startedOn": "2009-01-05T16:59:05.46+01:00",
  "finishedOn": "0001-01-01T00:00:00Z",
  "user": {
    "id": 848,
    "firstName": "John",
    "lastName": "Doe",
    "email": "john.doe@anonymous.com",
    "username": "johndoe",
    "interfaceLanguage": "English",
    "birthdate": "1987-05-28T00:00:00"
  },
  "publication": {
    "id": 5584,
    "name": "Test your assessmentQ knowledge",
    "description": "New publication",
    "type": "Assessment",
    "status": "Validated"
  },
  "components": [{
    "@odata.type": "#aQ.Api.Models.ComponentReport",
    "id": 48866,
    "name": "Component 1",
    "score": 0.0,
    "scoreMax": 11.0,
    "duration": 44,
    "type": "Component",
    "passed": false,
    "answer": null,
    "solution": null,
    "childItems": [{
      "id": 48868,
      "name": "Question 1",
      "score": 0.0,
      "scoreMax": 10.0,
      "duration": 34,
      "type": "OpenQuestion",
      "answer": {
        "value": "This is my answer."
      },
      "solution": {
        "value": "The model answer."
      },
    },
    {
      "id": 48869,
      "name": "Question 2",
      "score": 0.0,
      "scoreMax": 1.0,
      "duration": 10,
      "type": "MultipleChoiceSingleAnswerQuestion",
```

```

"answer": {
  "@odata.type": "#aQ.Api.Models.Answers.ChoiceAnswer",
  "choices": [
    {
      "value": "<p><span style=\"font-size:10px;color:#2E2E2E;\">Option A</span></p>",
      "score": 0.0,
      "selected": false,
      "correct": true
    },
    {
      "value": "<p><span style=\"font-size:10px;color:#2E2E2E;\">Option B</span></p>",
      "score": 0.0,
      "selected": false,
      "correct": true
    },
    {
      "value": "<p><span style=\"font-size:10px;color:#2E2E2E;\">Option C</span></p>",
      "score": 0.0,
      "selected": false,
      "correct": false
    },
    {
      "value": "<p><span style=\"font-size:10px;color:#2E2E2E;\">None of the above</span></p>",
      "score": 0.0,
      "selected": true,
      "correct": false
    }
  ]
},
"solution": {
  "@odata.type": "#aQ.Api.Models.Answers.ChoiceAnswer",
  "choices": [
    {
      "value": "<p><span style=\"font-size:10px;color:#2E2E2E;\">Option A</span></p>",
      "score": 0.0,
      "selected": false,
      "correct": true
    },
    {
      "value": "<p><span style=\"font-size:10px;color:#2E2E2E;\">Option B</span></p>",
      "score": 0.0,
      "selected": false,
      "correct": true
    },
    {
      "value": "<p><span style=\"font-size:10px;color:#2E2E2E;\">Option C</span></p>",
      "score": 0.0,
      "selected": false,
      "correct": true
    },
    {
      "value": "<p><span style=\"font-size:10px;color:#2E2E2E;\">None of the above</span></p>",
      "score": 1.0,
      "selected": true,
      "correct": true
    }
  ]
},
"childItems": []
},
{
  "id": 3829874,
  "name": "Recording exercise question",

```

```

    "score": 0.0,
    "scoreMax": 10.0,
    "duration": 192,
    "type": "RecordingQuestion",
    "answer": {
      "@odata.type": "#aQ.Api.Models.Answers.RecordingAnswer",
      "audio": "https://<channel_subdomain>.edumaticonline.com/teachandlearn/RetrieveBlob.ashx?
hashCode=F2CBABAD1A15D701688FE7B8DEEB471A&assetId=1101723",
      "video": null,
      "status": "Transcoded"
    },
    "solution": {
      "@odata.type": "#aQ.Api.Models.Answers.RecordingAnswer",
      "audio": null,
      "video": null,
      "status": null
    },
    "childItems": []
  }
}
}
}
}

```

The *answer* property will contain the answer entered/selected by the user whereas the *solution* property holds the correct answer for that particular question. The formatting of the answers and solutions will depend on the question type.

📘 Currently, only the details (answers & solutions) of questions of the following types will be returned:

- DragAndDropInColumnsQuestion
- DragAndDropTextInTextQuestion
- DropdownsInTextQuestion
- FillGapsInTextQuestion
- MultipleChoiceMultipleAnswersQuestion
- MultipleChoiceSingleAnswerQuestion
- MultipleChoiceMatrixQuestion
- OpenQuestion
- OrderVerticallyQuestion
- MatchingQuestion
- RecordingQuestion → including the option to download the recordings

For the other question types, the values will all be *null*.

4.4.2 Additional information with respect to processing answers

RecordingQuestion

In case of a recording exercise, the recording will be converted (transcoded) to a common file format. For an audio recording, and MP3 will be made available whereas for a video (webcam) recording an MP4 file will be created. The answer received through the assessmentQ REST API contains a status property indicating whether the recording has been transcoded or not. The possible values for this status are:

Status	Information
Original	Transcoding not started yet
Transcoding	Transcoding is ongoing

Status	Information
Transcoded	Indicating the answer (recording) is available for download
TranscodingFailed	Something went wrong while processing the recording - please contact Televic Education support

When the status is 'Transcoded', the 'audio' and 'video' properties in the answer will contain a download link to the audio and video track of the recording. Note that the video track will only be available in case of webcam recording. Furthermore, the video track will contain both audio and video together.

4.4.3 Results management for assignments of type 'scorm'


Assignments of type 'scorm' have permanent sessions. As a result, sessions of an assignment of type 'scorm' will always have the status 'incomplete'.

To obtain the results for sessions of an assignment of type 'scorm', you can filter on the assignment type 'scorm' and the FinishedOn value of the sessions. The FinishedOn value indicates when the last interaction of the user in the session took place.

You can schedule this call at a specific time each day (for example, every evening at 20:00) or with a certain frequency (for example, every two hours).

Example:

```
GET /v1/sessions?$filter=(finishedOn gt 2006-12-30T23:59:59.99Z and publication/type eq 'Scorm')
```

-  Please note that for assignments of type 'scorm':
- scores can only be obtained on the level of the assignment.
 - obtaining the answers and solutions is not supported for assignments of type 'scorm'.

4.4.4 Resthooks

We support two types of resthooks: **session finished** and **publish assignment results**.

Subscribing to a resthook can be done by sending a **POST** request to **/v2/subscriptions** with the following information:

- Name: the name under which you want to register this resthook. This name is also visible in the backoffice.
- EventType: the type of resthook you want to subscribe to, the two possible values you can enter here are: **SessionFinished** and **AssignmentResultsPublished**, alternatively the values '1' and '2' are also accepted.
- URL: the URL where we will be sending data when the rest hook is triggered
- SecurityKey (optional): extra security key for additional security (explained below)

Example body:

```
{
  "name": "My Web Hook",
  "eventType": "SessionFinished",
  "url": "https://televic-education.com/fetch",
  "securityKey": "MyVerySecretKey01"
}
```

Before a resthook is added, we send a test-request to the specific URL. This test-request is a **POST** request that contains an **empty body** and **"X-Hook-Secret"** header.

To validate your resthook, the response should be a **NoContent** response (statuscode 204) that uses the same header and header value as the request header.

So if the request contains a "X-Hook-Secret"-header with value "abc123", the response should also contain a "X-Hook-Secret"-header with value "abc123".

If you receive a 422 "BadEntity" response while trying to add a resthook, the supplied URL did not return the response we expected.

Session Finished

After registering a session finished resthook, every time a session get finished, this resthook will be called (**POST** request) with the following data:

```
{
  "payload": { "sessionId": 123 }
  "hook": { "id": 1, "target": "URL", "eventtype": "SessionFinished" }
}
```

With the sessionId, you can retrieve the session from the session-endpoint.

Assignment Results

The Assignment Results will be triggered every time someone in the backoffice publishes the results of an assignment.

The body of this request contains the following data:

```
{
  "payload": { "PublicationId": 123, "UserId": 456 }
  "hook": { "id": 2, "target": "URL", "eventtype": "AssignmentResultsPublished" }
}
```

The assignment id can then be used to fetch the assignment's data, the user id is the id of the user that published the results via our backoffice.

Security Key

If a security key was given while adding a resthook, all resthook requests will contain a **"X-hook-signature"**-header. The value of this header is the **payload** that has been hashed according to the HMACSHA256 algorithm.

The HMACSHA256 algorithm uses the security key as encryption key.

This header can be used to verify if the content that was sent by the server hasn't been tampered with.

Our users can check this by encrypting the payload they received via HMACSHA256 and the security key that was supplied when adding the resthook. If their encryption matches the value of the "X-hook-Signature"-header, they can be sure that the data is correct and safe to use.

For example:

A session finished resthook with payload SessionId = 123 and securityKey = abc => "X-Hook-Signature" = 17c5577856d43d0d2abbf6f6530b46f8093ce1f30be862919febdb7b49d61d7b

5 Exploring the assessmentQ REST API using Postman or curl

Postman and curl are two of the most widely tools for testing APIs. With the help of Postman and curl, it is possible to explore the assessmentQ REST API without the need for writing any line of source code.

In this section, we will provide some basic steps on how to use Postman and curl to start experimenting with the assessmentQ REST API. We will detail how to obtain an OAuth2.0 access token and use this token to obtain a list of all the users in assessmentQ.

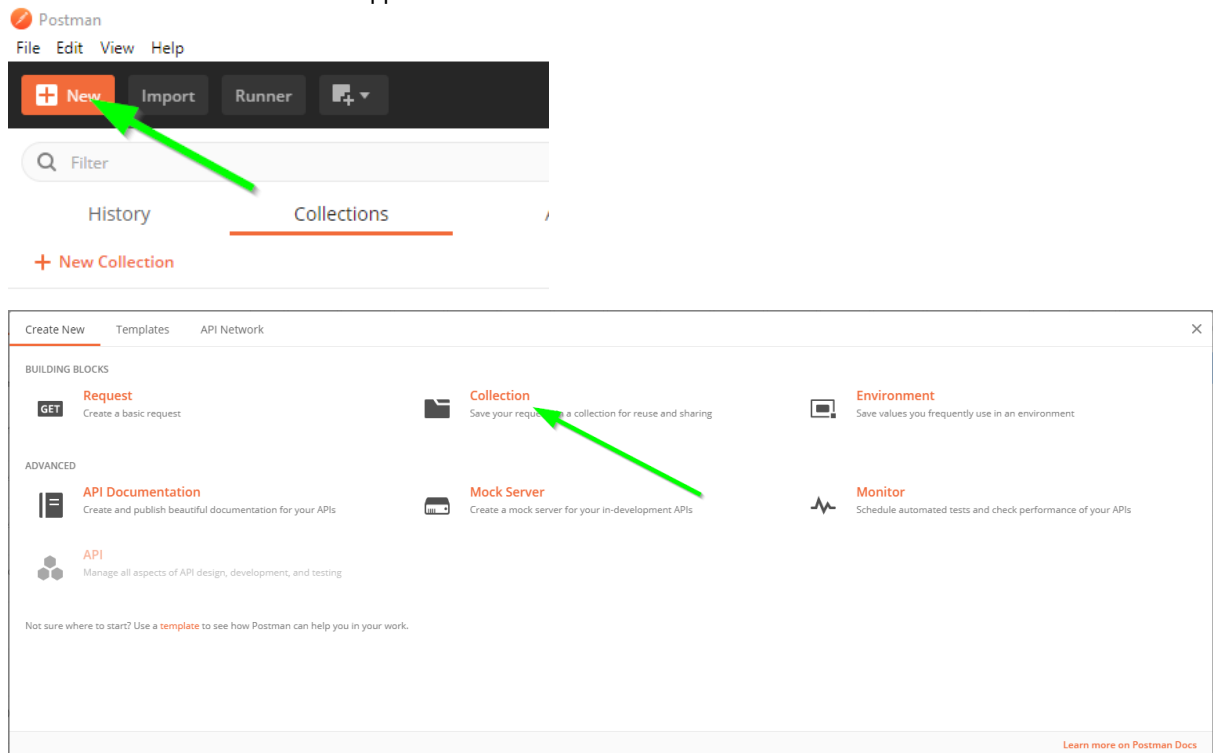
5.1 Postman

5.1.1 Initial configuration

Postman offers an intuitive UI to test and experiment with the assessmentQ REST API. The postman software can be downloaded directly from the postman website <https://www.postman.com/>.

Once you have installed the software, launch Postman and follow the steps below to test the assessmentQ REST API.

1. First you need to create a collection. This is a logical group of requests that you can organize into folders. So click the '+ New' button in the upper left corner and select 'Collection'.



2. In the 'Create a new collection' dialog, enter a name for the collection (e.g. assessmentQ REST API). Optionally, you can also provide a description.

CREATE A NEW COLLECTION

Name

assessmentQ REST API

Description Authorization Pre-request Scripts Tests Variables

This description will show in your collection's documentation, along with the descriptions of its folders and requests.

Make things easier for your teammates with a complete collection description.

- We will now add some variables to the collection which will facilitate authentication and using the assessmentQ REST API. Click the 'Variables' tab and complete the table as listed below.

CREATE A NEW COLLECTION
✕

Name

Description
Authorization
Pre-request Scripts
Tests
Variables

These variables are specific to this collection and its requests. [Learn more about collection variables.](#)

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	⋮	Persist All	Reset All
<input checked="" type="checkbox"/>	token_url	https://www.sign-in.educ ...	https://www.sign-in.education/e/connect/token			
<input checked="" type="checkbox"/>	client_id	████████████████████	████████████████████			
<input checked="" type="checkbox"/>	client_secret	████████████████████	████████████████████			
<input checked="" type="checkbox"/>	api_url	https://televic.api.assessr ...	https://televic.api.assessmentq.com			
	Add a new variable					

Example values. Please check the documentation for the real values you need to use

i Use variables to reuse values in different places. Work with the current value of a variable to prevent sharing sensitive values with your team. [Learn more about variable values](#) ✕

Cancel
Create

Replace the placeholder by the data received from Televic Education.

VARIABLE	INITIAL VALUE	CURRENT VALUE
token_url	<OAuth 2.0 Access Token URL>	<OAuth 2.0 Access Token URL>
client_id	<Client ID>	<Client ID>

VARIABLE	INITIAL VALUE	CURRENT VALUE
client_secret	<Client Secret>	<Client Secret>
api_url	<API Endpoint URL>	<API Endpoint URL>

4. Now click the create button to create the collection.

CREATE A NEW COLLECTION
✕

Name

Description
Authorization
Pre-request Scripts
Tests
Variables ●

These variables are specific to this collection and its requests. [Learn more about collection variables.](#)

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	⋮ Persist All Reset All
<input checked="" type="checkbox"/>	token_url	https://www.sign-in.educ ...	https://www.sign-in.education/e/connect/token	
<input checked="" type="checkbox"/>	client_id	[REDACTED]	[REDACTED]	
<input checked="" type="checkbox"/>	client_secret	[REDACTED]	[REDACTED]	
<input checked="" type="checkbox"/>	api_url	https://televic.api.assessr ...	https://televic.api.assessmentq.com	
	Add a new variable			

ⓘ Use variables to reuse values in different places. Work with the current value of a variable to prevent sharing sensitive values with your team. [Learn more about variable values](#) ✕

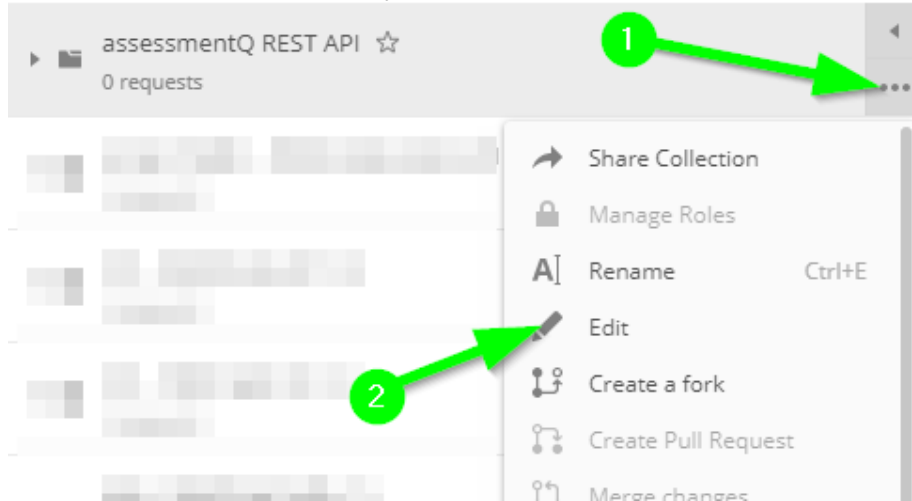
Cancel
Create



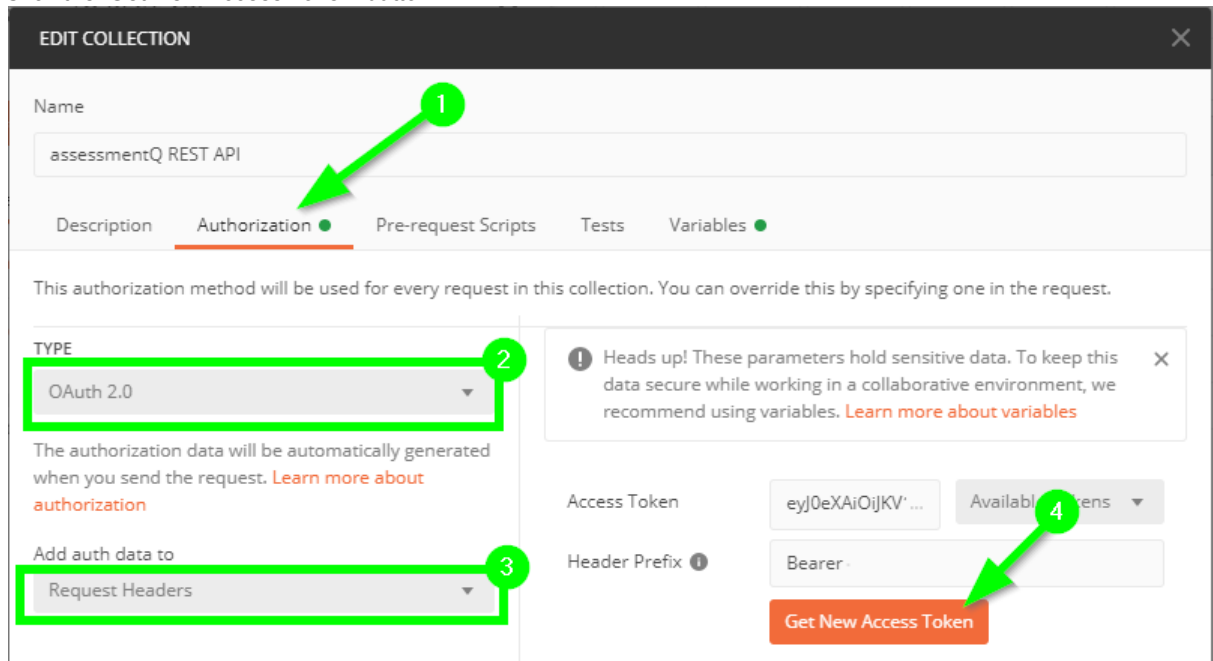
5.1.2 Setting up authorization

Now it's time to request an authorization token.

1. Click the three dots next to the newly created collection and select 'Edit' from the dropdown menu.



2. Go to the 'Authorization tab'.
Select 'OAuth 2.0' from the 'Type' dropdown.
Select 'Request Headers' from the 'Add auth data to' dropdown.
Click the 'Get New Access Token' button.



3. Enter a name for the token.
Select 'Client Credentials' from the 'Grant Type' dropdown and enter the variables as defined in step 4 (in the previous section) in the fields 'Access Token URL', 'Client ID', 'Client Secret'.
Note: you need to enclose the variables names between {{ and }} characters.
Enter 'assessmentQApi' (without the quotes) in the 'Scope' field and select 'Send as Basic Auth header' in the 'Client Authentication' dropdown.

Click 'Request Token' to request the authentication token.

GET NEW ACCESS TOKEN

Token Name: IdentityServer

Grant Type: Client Credentials

Access Token URL: {{token_url}}

Client ID: {{client_id}}

Client Secret: {{client_secret}}

Scope: assessmentQApi

Client Authentication: Send as Basic Auth header

Buttons: Cancel, Request Token


4. If the token has been requested successfully, a new dialog which will be presented. Click 'Use Token' to use the obtained token.

MANAGE ACCESS TOKENS ✕



All Tokens Delete ▾

IdentityServer

Token Details

Token Name IdentityServer 

Access Token eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IjR3JoN3haR2xqX1

(The Access Token value is partially obscured by a greyed-out area in the original image.)

5. Now finally click the 'Update' button to close the dialog.

EDIT COLLECTION

Name
assessmentQ REST API

Description Authorization Pre-request Scripts Tests Variables

This authorization method will be used for every request in this collection. You can override this by specifying one in the request.

TYPE
OAuth 2.0

The authorization data will be automatically generated when you send the request. [Learn more about authorization](#)

Add auth data to
Request Headers

Access Token eyJ0eXAiOiJKV1Q Available Tokens

Header Prefix Bearer

Get New Access Token

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

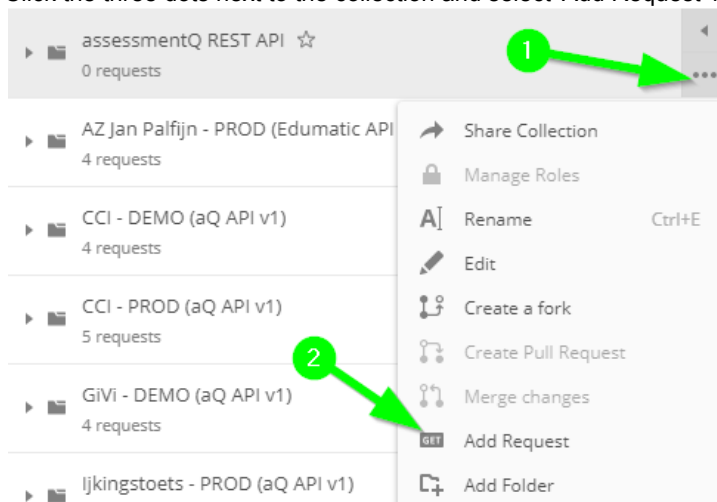
Cancel Update

Everything is now setup to add and execute an API call. In the next steps, we will demonstrate how to add a GET and POST api call.

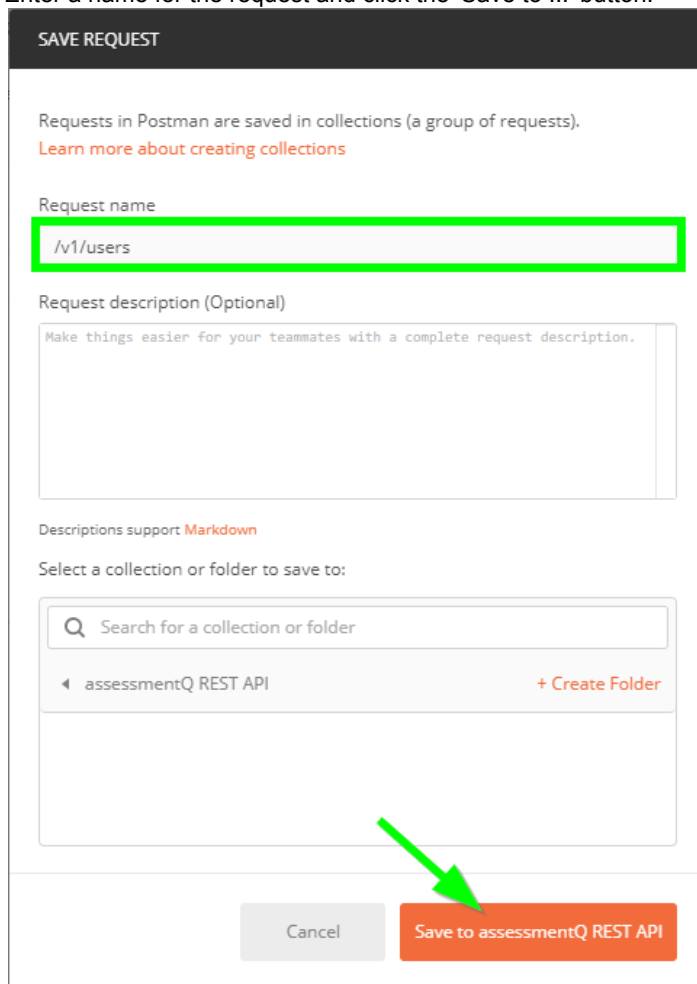
5.1.3 Adding a GET API call

Let's now add a simple GET call (e.g. /v1/users).

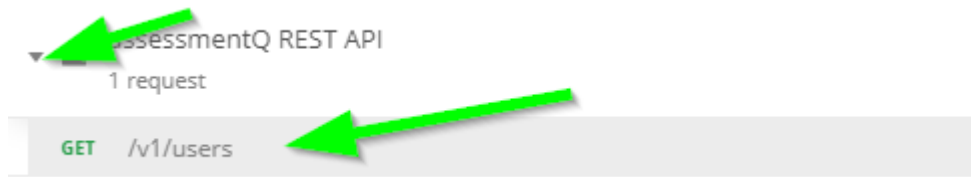
1. Click the three dots next to the collection and select 'Add Request' from the dropdown menu.



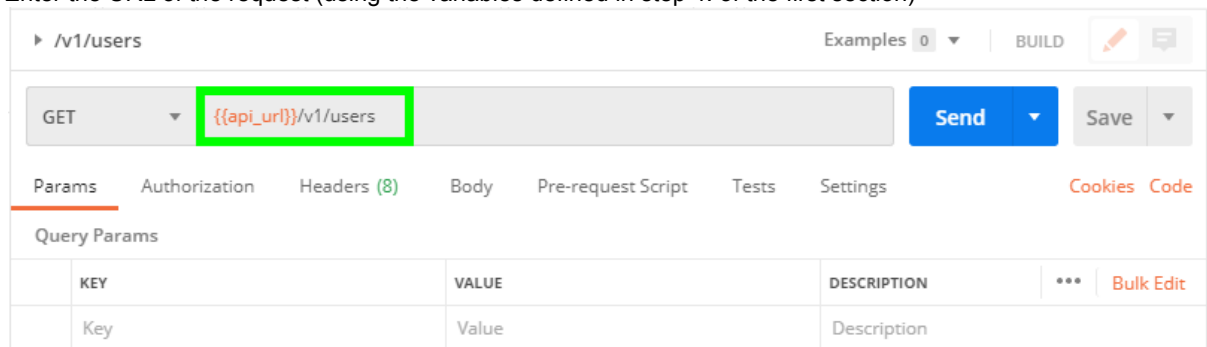
2. Enter a name for the request and click the 'Save to ...' button.



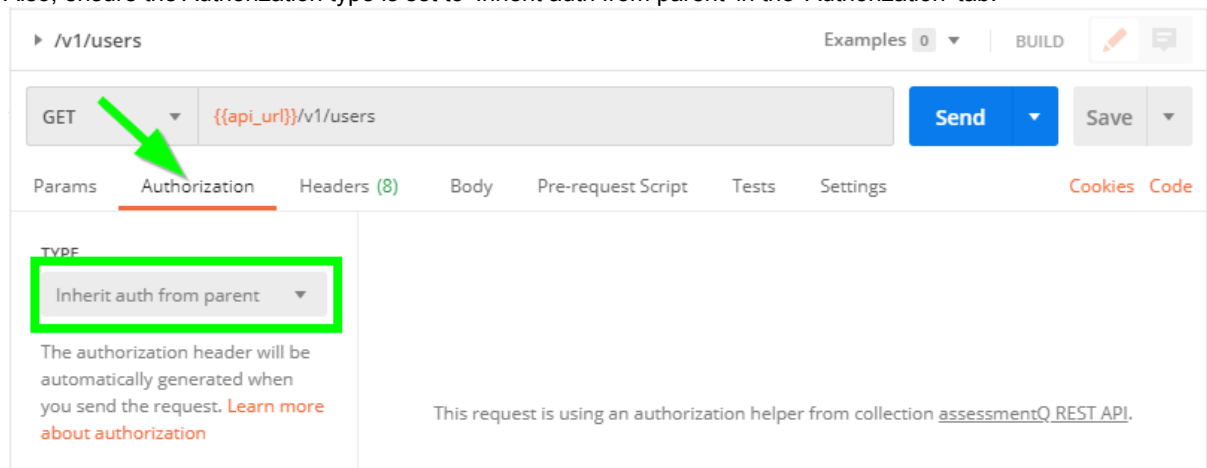
3. In the left part of the screen, expand the collection and click the added request.



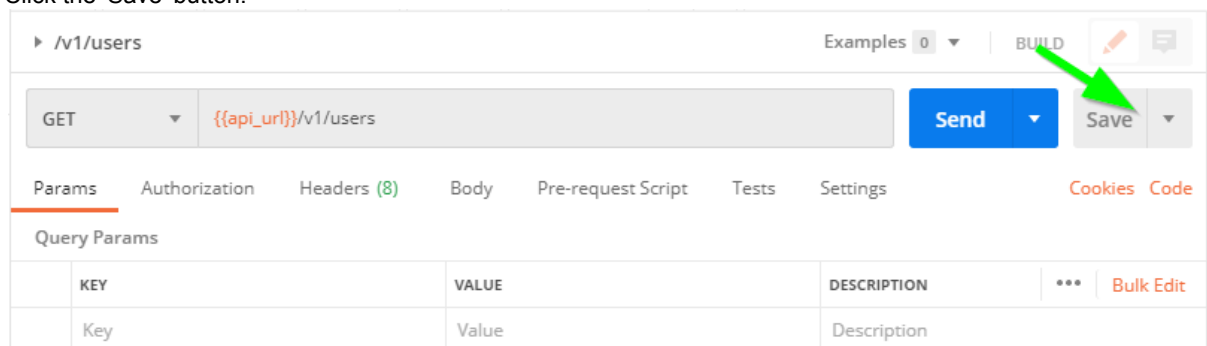
4. Now we can complete the details of the call in the right side of the window. Enter the URL of the request (using the variables defined in step 4. of the first section)



Also, ensure the Authorization type is set to 'Inherit auth from parent' in the 'Authorization' tab.



Click the 'Save' button.



- To execute the created request, click the 'Send' button. If the request completed successfully (check the HTTP response status code), you will see the returned data in the lower part of the screen.

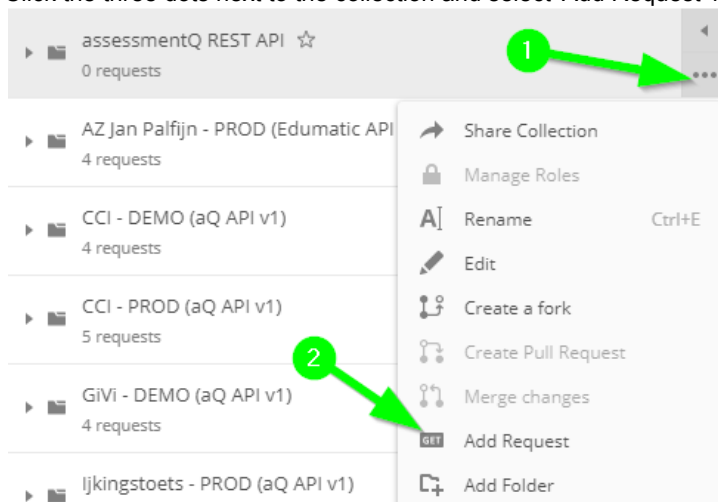
The screenshot shows a REST client interface for a request to `/v1/users`. The request method is `GET` and the URL is `{{api_url}}/v1/users`. The `Send` button is highlighted with a green arrow. The response status is `200 OK`, and the response time is `94 ms` and the size is `17.62 KB`. The returned data is highlighted with a green box and labeled `Returned data`.

```
17     "email": "s.joos@televic.edu",
18     "username": "c1200023",
19     "interfaceLanguage": "Dutch",
20     "birthdate": null
21   },
22   {
23     "id": 2356,
24     "firstName": "Dirk",
25     "lastName": "Verbeke",
26     "email": "d.verbeke@televic.com",
27     "username": "DV",
28     "interfaceLanguage": "English",
29     "birthdate": null
30   }
}
```

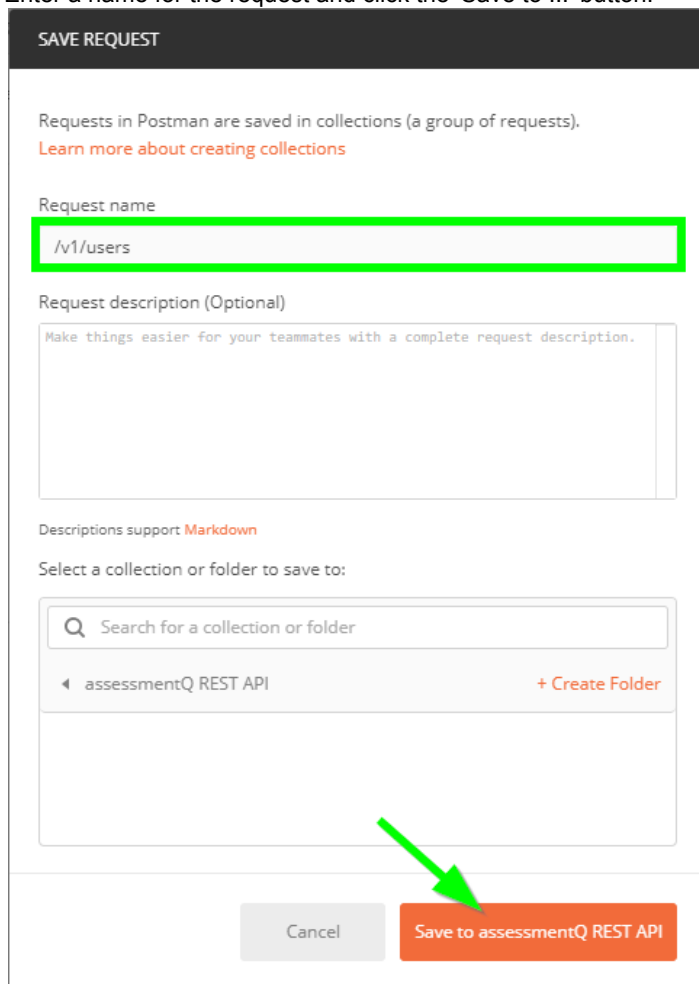
5.1.4 Adding a POST API call

Now we will demonstrate how to create a POST call (pushing data to the assessmentQ REST API).

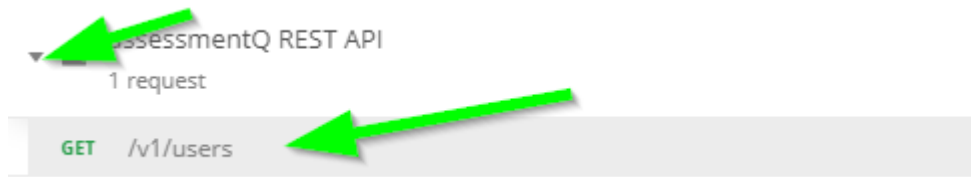
1. Click the three dots next to the collection and select 'Add Request' from the dropdown menu.



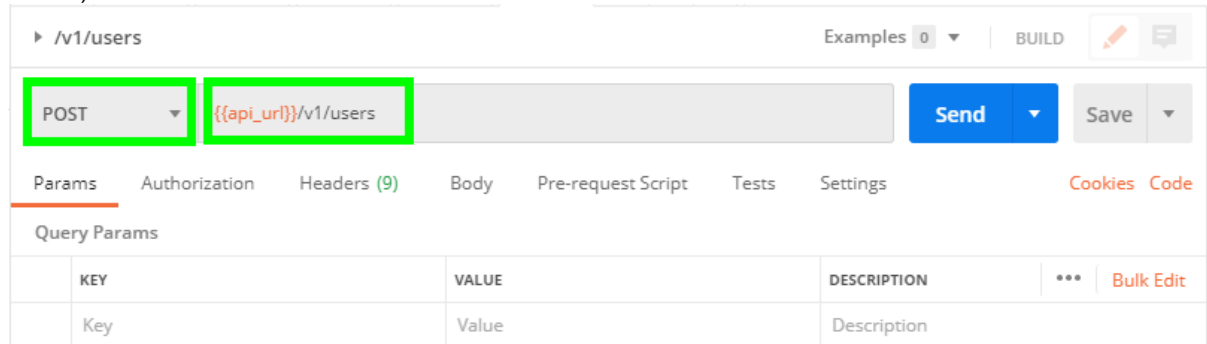
2. Enter a name for the request and click the 'Save to ...' button.



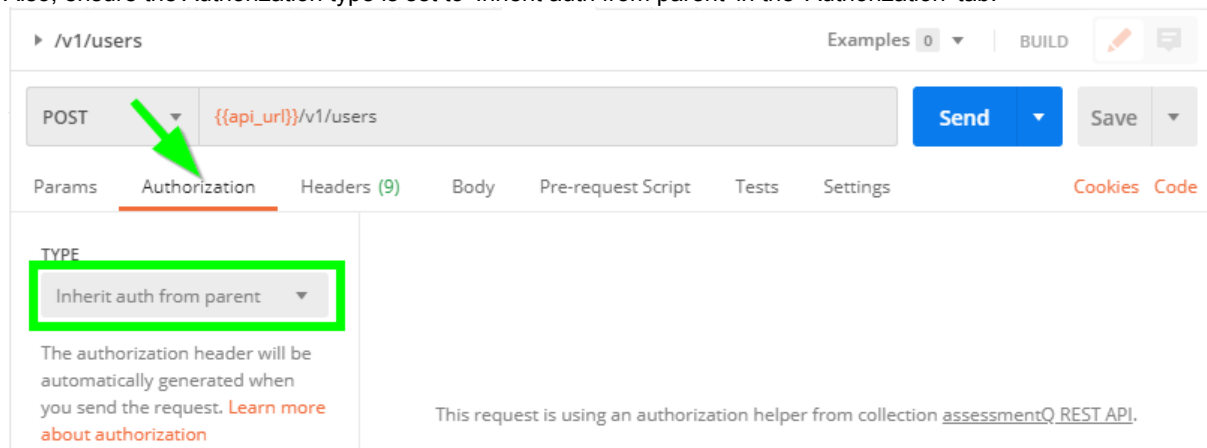
3. In the left part of the screen, expand the collection and click the added request.



4. Now we can complete the details of the call in the right side of the window. Set the method to 'POST' and enter the URL of the request (using the variables defined in step 4. of the first section)

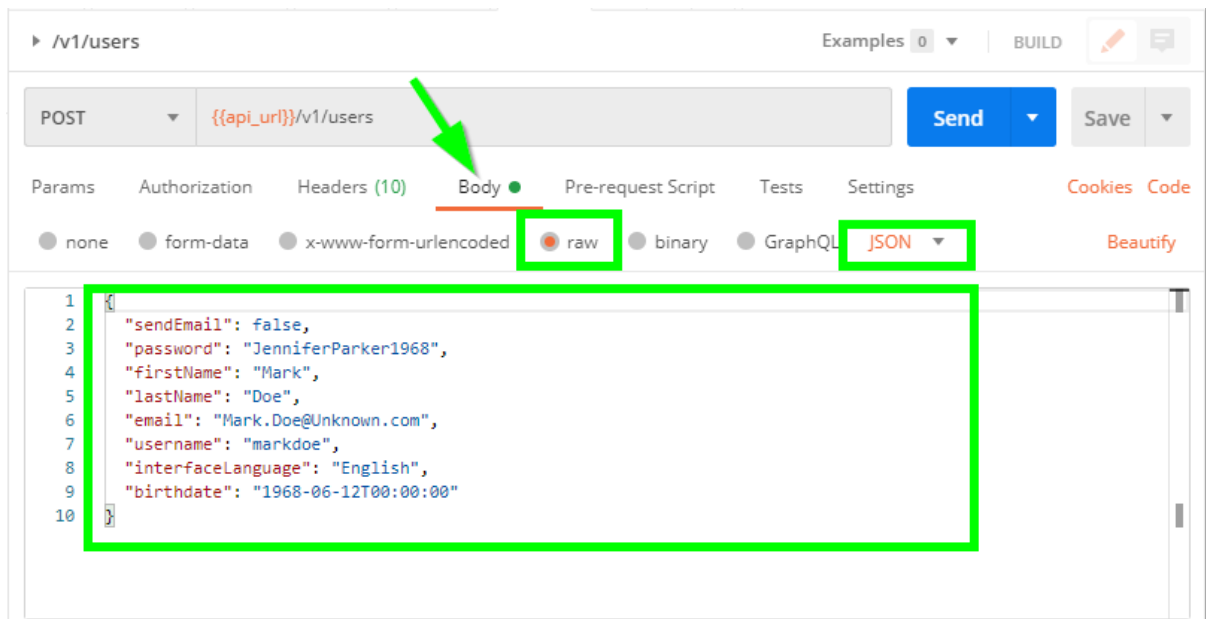


Also, ensure the Authorization type is set to 'Inherit auth from parent' in the 'Authorization' tab.

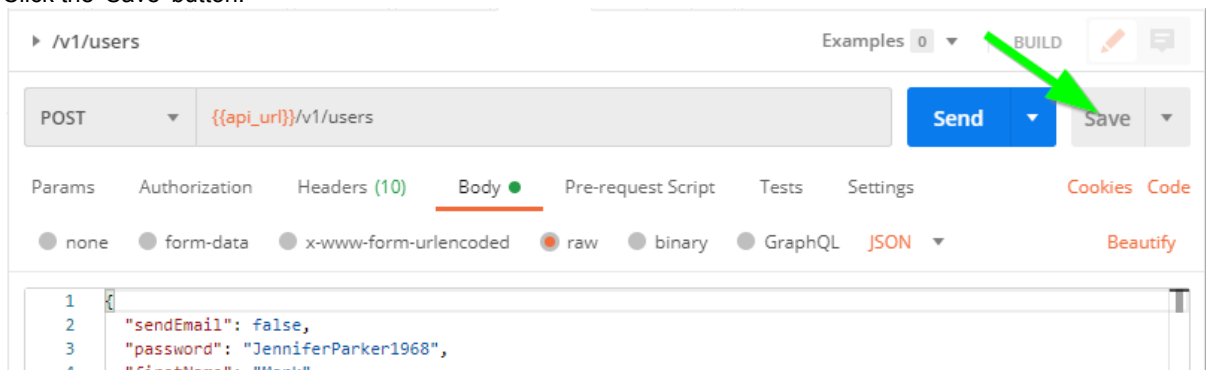


Now we need to enter the data we want to send to the assessmentQ REST API. As detailed in the documentation, all data is formatted as JSON. So navigate to the 'Body' tab and indicate that we will be sending raw JSON data.

Next, enter the JSON data in the texarea below.



Click the 'Save' button.



- To execute the created request, click the 'Send' button. If the request completed successfully (check the HTTP response status code), you will see the returned data in the lower part of the screen.

The screenshot shows the Postman interface for a POST request to `/{api_url}/v1/users`. The request body is a JSON object with the following fields:

```
1 {
2   "sendEmail": false,
3   "password": "JenniferParker1968",
4   "firstName": "Mark",
5   "lastName": "Doe",
6   "email": "Mark.Doe@Unknown.com",
7   "username": "markdoe",
8   "interfaceLanguage": "English",
9   "birthdate": "1968-06-12T00:00:00"
10 }
```

The response status code is **201 Created**. The returned data is as follows:

```
1 {
2   "@odata.context": "https://televic.api.assessmentq.com/v1/$metadata#Users/$entity",
3   "id": 798982,
4   "firstName": "Mark",
5   "lastName": "Doe",
6   "email": "Mark.Doe@Unknown.com",
7   "username": "markdoe",
8   "interfaceLanguage": "English",
9   "birthdate": "1968-06-12T00:00:00+02:00"
10 }
```

5.1.5 Additional information

Note that the authentication token remains valid for 10 minutes. In order to request/renew a token, follow steps as detailed in the section 'Setting up authorization'. In this case, the variables will already be filled in (in step 7).

For more information on Postman, please check their [Learning Center](#).

5.2 Curl

To obtain an access token using curl, executing the following command:

```
curl --request POST --data "grant_type=client_credentials&client_id=<YOUR CLIENT ID>&client_secret=<YOUR CLIENT SECRET>&scope=assessmentQApi" <ACCESS TOKEN URL>
```

Replace <YOUR CLIENT ID>, <YOUR CLIENT SECRET>, and <ACCESS TOKEN URL> by the information provided at the beginning of this document.

If the call completes successfully, an access token will be returned:

```
{
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6I1JqR3JoN3haR2xq",
  "expires_in": 600,
  "token_type": "Bearer"
}
```

The value of the "access_token" property contains the access token that needs to be submitted to the API requests. Note that for security reasons, we obfuscated the access token in the example above. In reality, the access token value will be much longer.

Once the access token has been copied, it can be supplied to the API call for, for example, querying the list of users known in assessmentQ:

```
curl --request GET --header "Authorization: Bearer <ACCESS TOKEN>" <API ENDPOINT URL>/v1/users
```

Replace <ACCESS TOKEN> by the token returned by the previous command (in this example case *eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6I1JqR3JoN3haR2xq*) and substitute <API ENDPOINT URL> using the information provided at the beginning of this document.

5.3 Node.js

Node.js code example.

```
1  const https = require("https");
2
3  const client_id = "[Your API Key]"; // for example: API-EDU-XXXXXXXXXX
4  const client_secret = "[ClientSecret]"; // for example: "XF1wDgJAckdb74t<EvpJVix0jUIIdzm*ez)r!hnZ(0eoV#"
5  const identity_url = "[Url of the assessmentq authentication provider]"; // for example: "https://demo.sign-
  in.education"
6  const api_url = "[https://<customername>.api.<environment>.assessmentq.com]"; // for example: "https://
  televic.api.demo.assessmentq.com" or "https://televic.api.assessmentq.com"
7
8  main();
9  function main() {
10     getAccessToken().then((token) => {
11         getPublications(token).then((result) => console.log(result));
12     });
13 }
14
15 function getAccessToken() {
16     return new Promise((resolve, reject) => {
17         const data = `grant_type=client_credentials&client_id=${client_id}&client_secret=${client_secret}
  &scope=assessmentQApi`;
18         const options = {
19             hostname: identity_url.replace("https://", ""),
20             port: 443,
21             path: "/e/connect/token",
22             method: "POST",
23             headers: {
24                 "Content-Type": "application/x-www-form-urlencoded",
```

```

25         "Content-Length": data.length
26     }
27 };
28
29     const req = https.request(options, res => {
30         res.on("error", error => reject(error));
31         res.on("data", d => {
32             const resultObject = JSON.parse(d);
33             resolve(resultObject["access_token"]);
34         });
35     });
36
37     req.on("error", error => reject(error));
38     req.write(data);
39     req.end();
40 });
41 }
42
43 function getPublications(token) {
44     return new Promise((resolve, reject) => {
45         const options = {
46             hostname: api_url.replace("https://", ""),
47             port: 443,
48             path: "/v1/publications",
49             method: "GET",
50             headers: {
51                 Authorization: "Bearer " + token
52             }
53         };
54
55         const req = https.request(options, res => {
56             res.on("error", error => reject(error));
57             res.on("data", d => resolve(String(d)));
58         });
59
60         req.on("error", error => reject(error));
61         req.end();
62     });
63 }

```

6 Rate limiting

The number of requests per interval is limited. Detailed information on the rate limiting can be found in specific headers which will be sent on each request.

Header	Description
X-Rate-Limit-Limit	The length of the rate limit period (eg. 1m, 12h, 1d).

Header	Description
X-Rate-Limit-Remaining	The number of requests remaining until the reset.
X-Rate-Limit-Reset	UTC date time (ISO 8601) when the limits resets.
Retry-After	The number of seconds remaining in the current period during which no requests can be handled by the API. This header is only present when the rate limit has been reached.

When the ratelimit has been reached, all requests will return with a HTTP statuscode 429 and ReasonPhrase 'API calls quota exceeded!'.

7 Contact information

For any further information on the assessmentQ REST API or in case additional support is needed, please contact the Televic Education helpdesk via <http://support.televic-education.com> or +32 51 79 14 98.