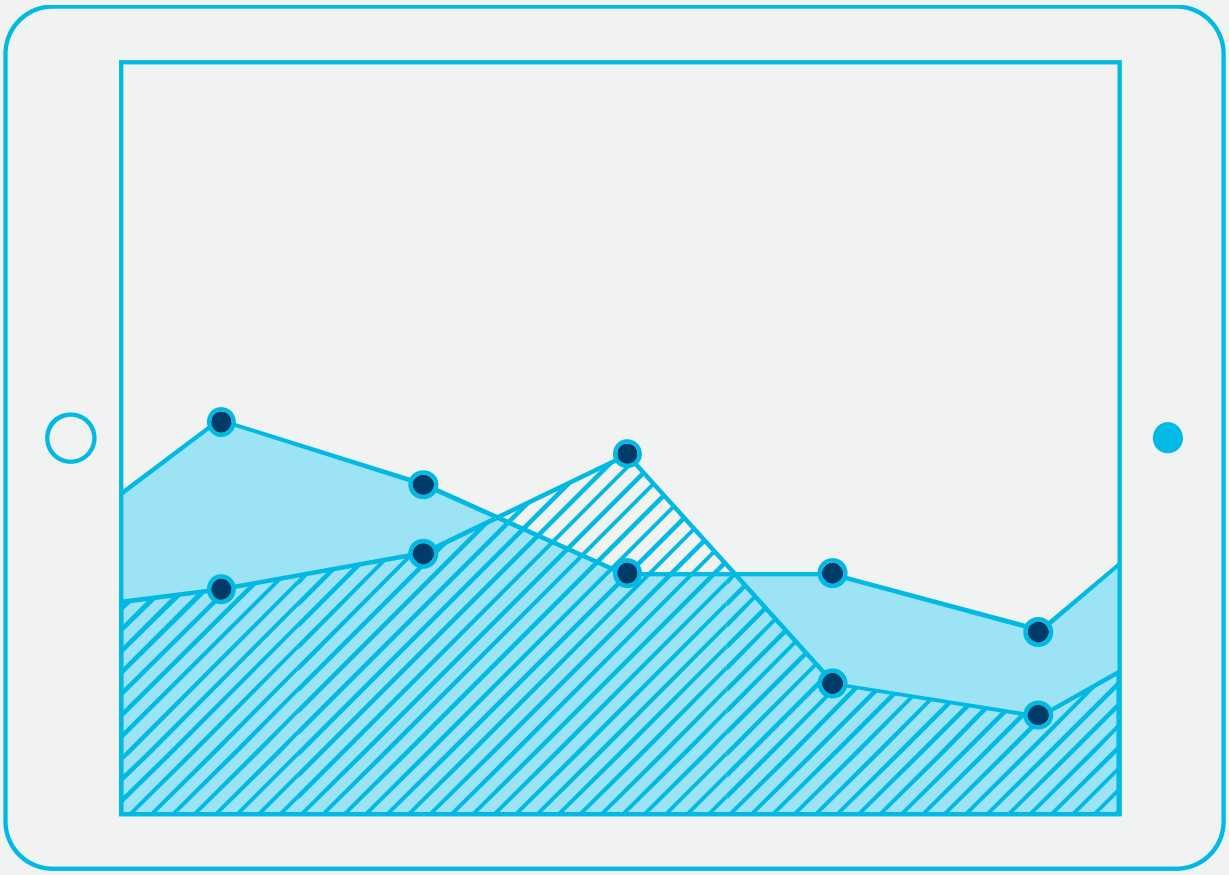
CheckWare®



CheckWare Client API

Technical implementation guide

Table of contents

Introduction to Client API

The CheckWare Client API is an extension to the CheckWare data collection platform which enables APP developers to securely store data into the CheckWare platform.

General implementation

The Client API is a REST based web API. It currently allows building external APP's for Respondents. (Future releases will extend respondent capabilities and add the possibility to build clinician APP's as well).

APP ID and APP Secret

To be able to add an APP you need to register (contact CheckWare) to get hold of an APP-ID and an APP secret. These are static values that do not change over time (unless there is a need for making a new secret if previous secret is compromised).

The ID will be used in CheckWare to represent the APP. The secret needs to be included in the APP code as a "proof" of a being the actual APP (In future releases this will be extended with full oAuth2 process).

Access token

When an end-user (respondent) want to connect an APP there is an initialization stage that is done once, which generates a connection access token. This is a unique code that tells the CheckWare Client API which respondent and which treatment the respondent's APP is connected to.

This is conveniently stored in a QR code for scanning by APP.

Client API root URL

When the end user connects the APP to CheckWare the root URL for this specific CheckWare installation is exposed together with the access token.

This is conveniently stored in a QR code for scanning by APP.

Request headers

Every single request towards the API MUST HAVE these http headers included:

HTTP Header	Description	Example
Authorization	The access token provided by the user interface in CheckWare.	896lq22on4gc8ck88c0sckk3w48swgs4swoqc84sks8g4gsco
AppSecret	The secret provided by CheckWare after registering the APP into the system.	a9687hj4359gnnTTQ6Q3v6y48B7MccQ6AMS4764b65n5zQ3dui

Any request missing these headers will not be processed.

If any of the values are incorrect, the client API will return an error message which includes a technical reason why.

Client API – Respondent

You will need to add this to the client API root URL. See the API reference guide for details in the requests (Currently not available). This documentation only elaborates on the purpose of the methods.

GET /assessment-instance/

Get a list of all assessments including data for the connected respondent's treatment. This includes assessments in the past and in the future. It also includes information about the status of the assessment instance.

This list includes all assessments, also those not added by the APP itself.

POST /assessment-instance/

Whenever your APP needs to store information securely into the CheckWare platform this is done by posting an assessment instance with the collected values. These data will be stored in CheckWare and will be represented through an assessment (A questionnaire form).

CheckWare is designed in a way that every single POST should represent a single measurement (or recording of some sort).

Required changes in APP

Connection preferences

AS an APP developer you need to make a small change to the APP to store two vital pieces of information: Client API root URL and the Access token.

This is provided by the CheckWare user interface and is conveniently inserted into a QR code as well to avoid manual typing of these rather complex tasks.

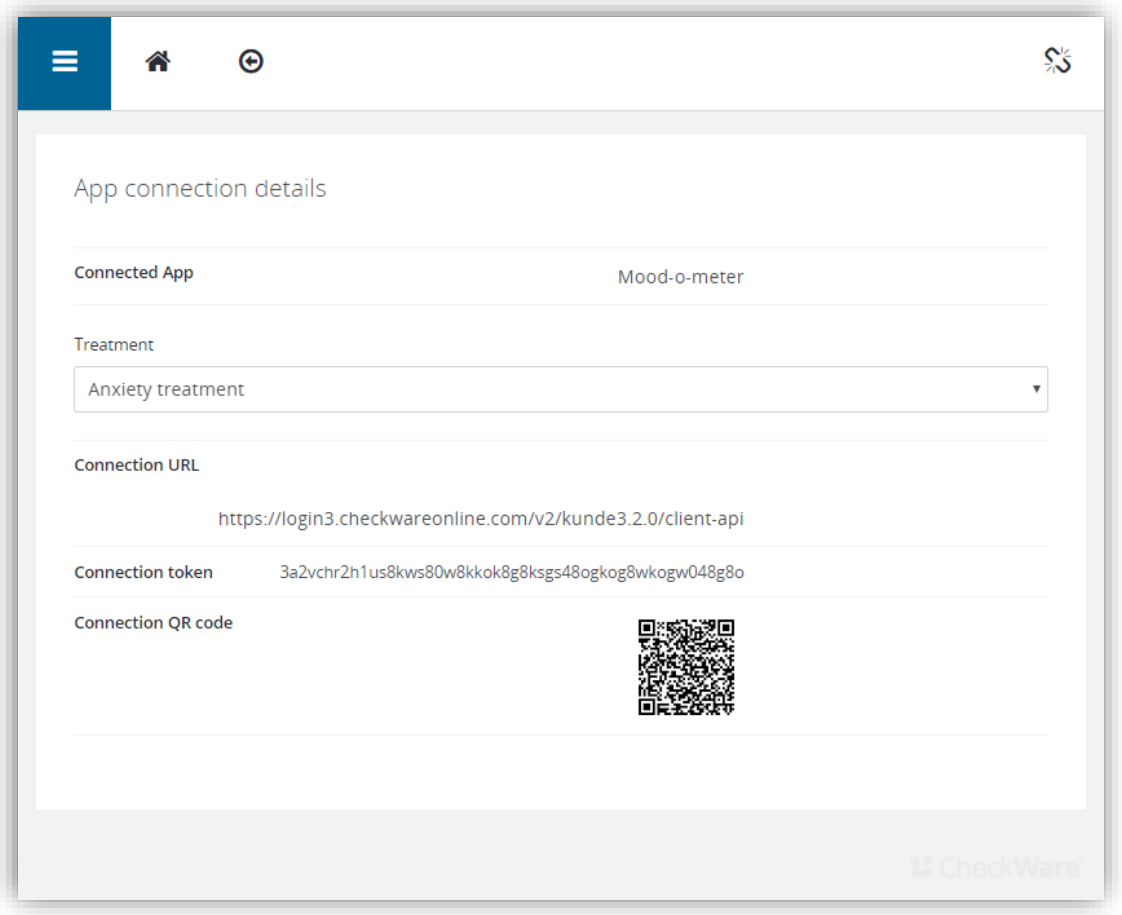


Illustration: Example of the web page in CheckWare where respondent get access to connection details

Your APP needs to be modified to insert this information into the APP store this information somehow on the device as it should only be done once.

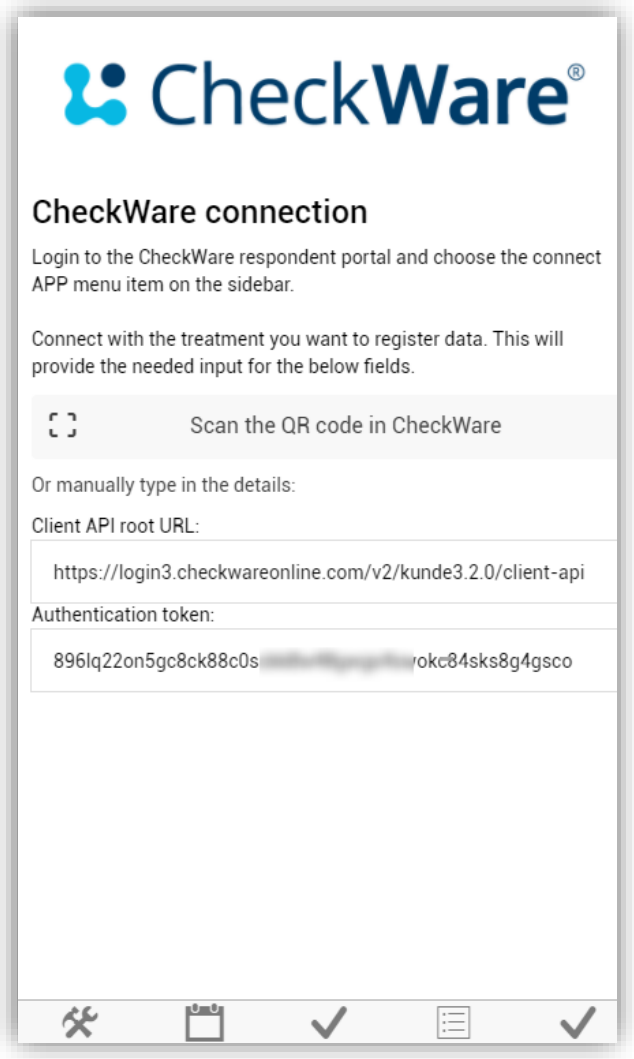


Illustration: Example of an APP settings page where this information is stored.

Data collection and sending

The APP usually contains some sort of interaction to collect data that you want to safely store in the CheckWare server, and at the same time make available to Clinicians for further observations. Reports made in CheckWare based on the data that is passed into the system will be automatically generated and transported into connected HER systems etc.

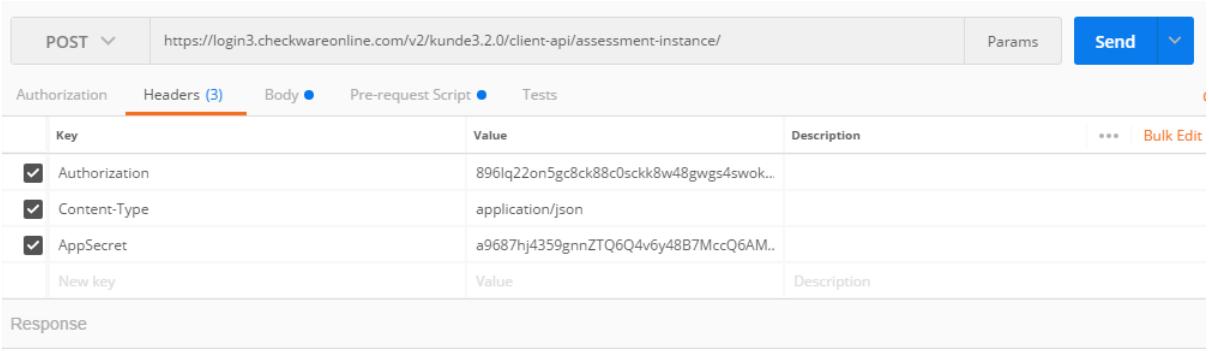
The data that are collected and passed to CheckWare will also be available for export into SPSS and other data analysis tools.

Passing the data to CheckWare is a simple JSON REST POST request.

Examples of request

HTTP header example

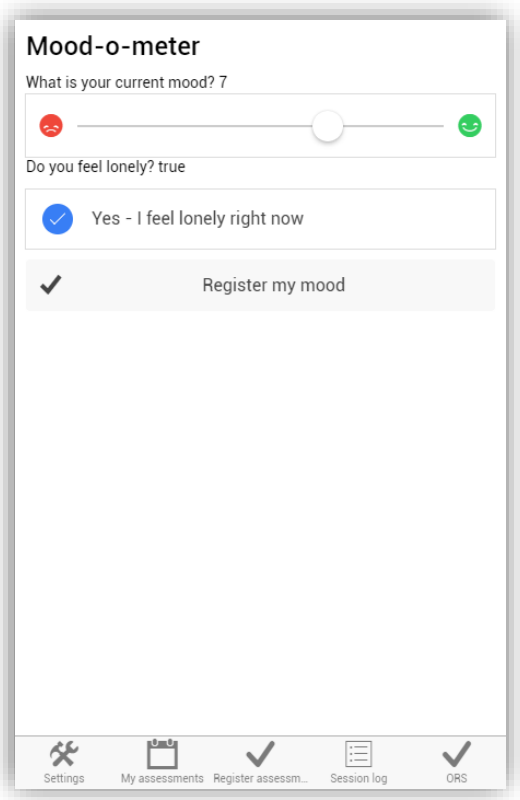
Snippets from postman example on posting an assessment instance:



Regardless of which data REST method you choose to invoke the header is always the same.

HTTP POST body example

The body is specific for each APP as it represents the data the APP collects, but the structure is always the same.



In our example we have a data interface where we collect a mood on a scale from 0-10 in addition to a simple question about being lonely which is a Boolean true or false.

IF the respondent ticks the lonely box we will trigger a flag in the CheckWare system for special followup.

We have a calculation which is the sum of all slider values (= the same value as the slider since we only have one slider in the example)

Based on these input we need to construct an HTTP POST request with a body reflecting the collected data and the algorithms we have defined:

```
{
  "assessment":{
    "code":"cw-mood-o-meter"

  },
  "answer":[{
    "code":"MOOD","value":[7]
  }, {
    "code":"LONELY","value":[1]
  }],
  "calculation":[{
    "code":"MOOD_INDEX","value":7

  }],
  "flag":[{
    "code":"LONELY_FLAG","value":true

  }],
  "startAt":"2017-10-18T11:48:32+00:00",
  "endAt":"2017-10-18T11:48:32+00:00",
  "modifiedAt":"2017-10-18T11:48:32+00:00",
  "submittedAt":"2017-10-18T11:48:32+00:00"

}
```

Let's look at the body part by part to understand the details.

First we have to tell the system which assessment (questionnaire) should be used for this data post. Usually CheckWare will assist in building this together with the APP developer.

In this example there is an assessment with the code "cw-mood-o-meter"

```
{
  "assessment":{
    "code":"cw-mood-o-meter"

  },
```

Next we need to add the values that are collected. Usually CheckWare will create an assessment in the system that will work even if added manually in the CheckWare system as well. If possible form widgets that looks the same as the ones in the APP will be used to ease the process of making it look the same.

In our example we will post the answer to the two questions (slider and checkbox). This will look this:

```
"answer":[{
  "code":"MOOD","value":[7]
}, {
  "code":"LONELY","value":[1]
}],
```

Notice that the value in these examples are potentially an array [ ]. This is because the slider in CheckWare is a slider of a defined set of options which in the CheckWare system could contain more than one value (if used as a range). The same goes with the checkbox. In this case it is only one option, but it could in theory be multiple values.

If this was let's say represented as a numeric value instead the value would be passed into CheckWare as a single value (without the [ ]).

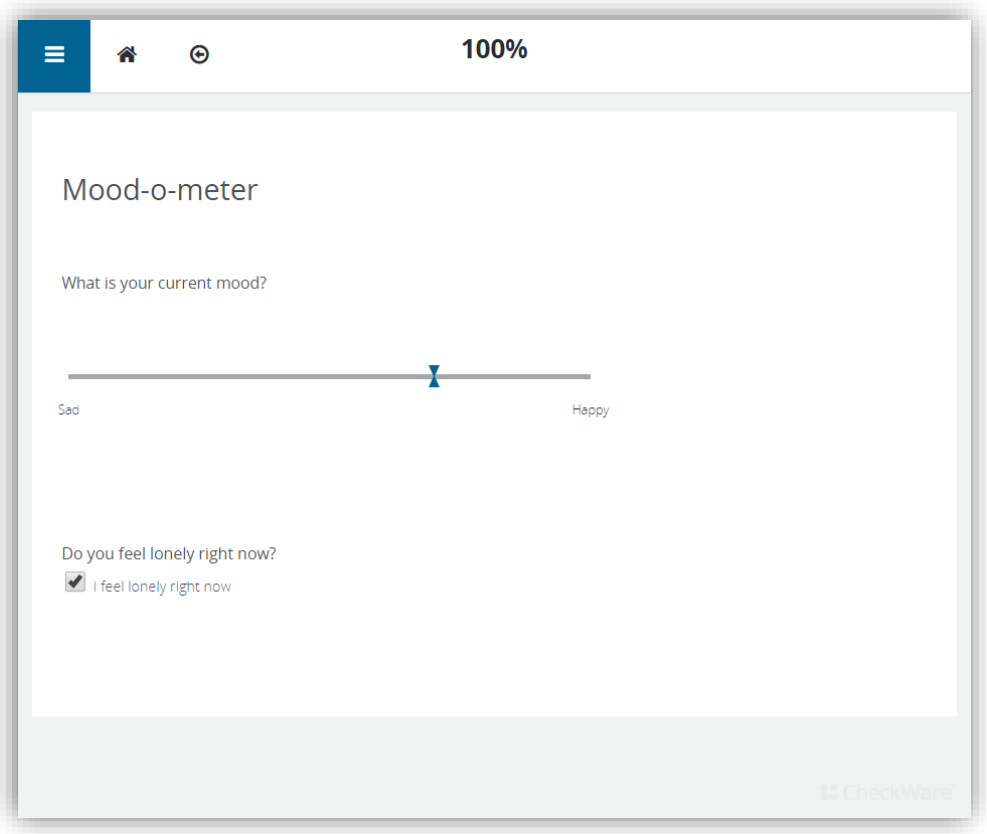


Illustration: This is how the data looks like if opened by a clinician in CheckWare

In addition to the answers which normally are visualized as part of the questionnaire in CheckWare you can pass calculation values to the system as well. These are not visually present in CheckWare but are available in data exports and are often used as elements to build plots and graphs in the assessment reports.

In our example we have a very simple calculation which represent the sum of the sliders. We only have one slider an d the value was 7, so the calculation is 7.

```
  "calculation":[{
    "code":"MOOD_INDEX","value":7

 }],
```

In our algorithms we have defined a special state which we want to flag in the CheckWare systems for special treatment. This is referred to as a flag (Not all assessments have this).

Since the checkbox of being lonely was ticked, we want to flag this into the system:

```
  "flag":[{
    "code":"LONELY_FLAG","value":true

  }],
```

Notice that when a flag is defined you must pass the flag data as either true or false. You cannot skip sending the flag if it is false.

The final thing we want to do set the dates for when the data was collected. There are four important date and time attributes:

```
  "startAt":"2017-10-18T11:48:32+00:00",
  "endAt":"2017-10-18T11:48:32+00:00",
  "modifiedAt":"2017-10-18T11:48:32+00:00",
  "submittedAt":"2017-10-18T11:48:32+00:00"
```

startAt represents the time when the user was able to start entering data.

endAt represent the time when an incomplete dataset was considered not completed (expire time).

modifiedAt represent the time when the data was last modified by the user.

submittedAt represent the time when the user actively clicked the “Send” button or in other means decided that this measurement is now complete. Notice that this does not have to be the time when the data is passed to CheckWare. This will allow an APP developer to create off-line capabilities, but still keep the actual time of collection to be sent to the CheckWare system when the APP is back online and syncs data with CheckWare.