# 1 OneQA Integration

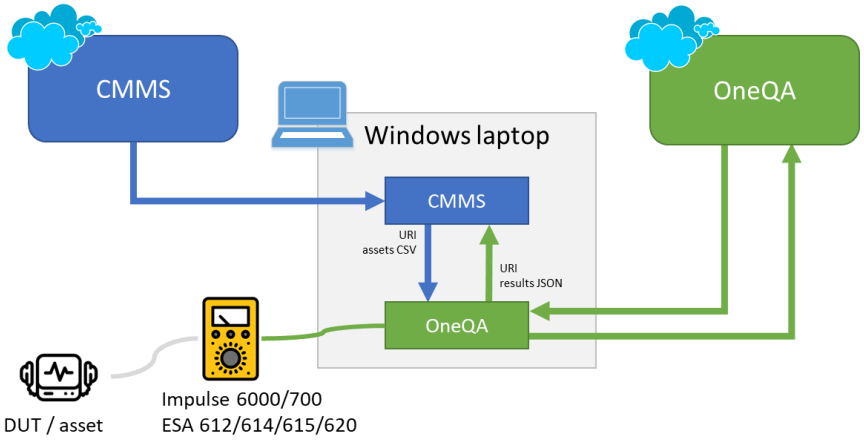
Typically integration would be done with an existing CMMS to carry out the asset quality control part of workorders.

There are 3 main objectives with integration of OneQA in to some existing tools or workflow.

1. **Synchronize asset information**
   Make OneQA aware of what to control and include information about the asset in results
2. **Run procedure**
   Increase efficiency for user to start a control and make sure the correct procedure is used
3. **Report results**
   Increase efficiency for user and make sure results and asset/work status gets updated correct

## 1.1 Operational Scenario

OneQA is a Windows application with an online cloud infrastructure to store all data. OneQA will be installed on a Windows laptop to be used by the biomed engineer to do the tests.

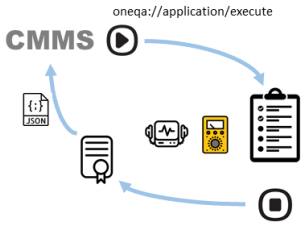



## 1.2 Typical Workflow

They typical workflow will include some external CMMS solution that keep track of the assets and generate workorders that include some type of quality control tasks to be carried out.

1. Synchronize assets from CMMS to OneQA, export import CSV list


2. Process workorders
   a. Pick up workorder in CMMS
   b. Start procedure in OneQA, CMMS provide URI link
   c. Follow the procedure in OneQA
   d. Report the result back to CMMS, OneQA provide JSON result



### 1.2.1 Asset Synchronization

Asset synchronization must be carried out manually by exporting a list of assets from the external application and imported in to OneQA. See OneQA documentation for more information.

### 1.2.2 Start Procedure

A procedure can be started manually inside OneQA or by an external application making a URI call in to OneQA on the local computer. See this document for more information about URI call to OneQA

### 1.2.3 Report Result

There are 3 main options to report the result from OneQA. The different ways of reporting the results can be combined and customized dependent on the specific workflow needs.


- User save a result as PDF from inside OneQA. The PDF will by default be formatted as the result but can also be customized using templates. For more information see OneQA documentation.


- User save a result as CSV from inside OneQA. The CSV file will contain all information from the result. For more information see OneQA documentation.


- OneQA automatically save/send the result as JSON when the result is done/approved. The JSON file will contain all information from the result. For more information see this document.

# 2 OneQA Command Line Interface

The OneQA Command Line Interface (CLI) is a way to integrate OneQA with existing tools and workflows. Interacting with OneQA from the command line it is possible to…

## 2.1 Call in to OneQA

Its possible to interact with OneQA by call in using a command line URI structure. Calling OneQA will interact with the already running instance of OneQA or start a new one if not already existing. The call is done using Windows URI schema for calling application and parameters can be supplied when needed.

To test this functionality of OneQA a web browser can be used. When a oneqa:// URI is given the browser should open the application according to the URI with parameters given.

For the call to work the user must be signed in to the tenant where any of the referenced asset/procedure/result is located. If user is currently signed in to other tenant the call will simply fail as referenced asset/procedure/result will not be found.

As the URI used to call in to OneQA can contain reserved characters it must be percent encoded before used. This encoding should be done on the part added after ? in the call

For more information see: https://en.wikipedia.org/wiki/Percent-encoding

### 2.1.1 Navigate To

There are some navigation calls that can be used to take the UI to specified views inside OneQA. This can be used for linking the user in to OneQA from some external application.

| Call | Navigate To |
|---|---|
| `oneqa://application/assets` | List of all assets |
| `oneqa://application/asset/{id}` | Show specific asset<br>{id} GUID to procedure (currently not available) |
| `oneqa://application/procedures` | List of all procedures |
| `oneqa://application/procedure/{id}` | Show specific procedure<br>{id} GUID to procedure (from “Copy procedure URI” in application) |
| `oneqa://application/results` | List of all results |
| `oneqa://application/results/{id}` | Show specific result<br>{id} GUID to result (from result CSV or JSON export) |

There is no error handling for the navigate to call. If the supplied {id} is invalid or not existing the call will fail and no navigation action is carried out.

### 2.1.2 Run Procedure

Its possible to run a procedure inside OneQA using a URI. This feature would be used to link a user in to a started procedure in OneQA.

```
oneqa://application/execute?procedureId=procedureid
```

```
oneqa://application/execute?procedureId=procedureid&version=1
```

```
oneqa://application/execute?procedureId=procedureid&assetId=assetid
```

```
oneqa://application/execute?procedureId=procedureid&
&callbackType=sharedStorage&callback=callbackuri
```

- procedureId GUID to procedure (from “Copy procedure URI” in application)
- assetId OPTIONAL Asset ID as given (copy from Asset ID property in application)
- version OPTIONAL version to run (fetch from version information in application)
  If no version is given the latest version of the procedure will be used
- callbackType OPTIONAL used to let OneQA call other application when procedure is done see section ***Error! Reference source not found. Error! Reference source not found.*** for further information.
- callback OPTIONAL used to let OneQA call other application when procedure is done see section ***Error! Reference source not found. Error! Reference source not found.*** for further information.

**Example:**

```
oneqa://application/execute?procedureId=d826477c-fe78-4cc2-9715-c1e22c578b67

oneqa://application/execute?procedureId=d826477c-fe78-4cc2-9715-c1e22c578b67&version=2

oneqa://application/execute?procedureId=d826477c-fe78-4cc2-9715-c1e22c578b67&assetId=123-abc
```

**Error handling:**

If the procedure GUID do not exist for the tenant user is currently signed in for the call will fail.

If the version of procedure do not exist the call will fail.

If the supplied assetId do not exist procedure will start but user get an error message in OneQA that assetId do not exist. In this case no assetId will be added to the running procedure.

#### 2.1.2.1 Settings

Parameters can be supplied in the format of

```
settings.setting-type=json-value
```

- setting-type the setting to supply
- json-value setting parameter value in json format

##### 2.1.2.1.1 Work Order ID

Supplies the work order id in URI to auto fill in the procedure.

**setting-type:**

```
settings.WorkOrderId=json-value
```

**json-value:**

```
{
    "id": "workorderid"
}
```

- “id” string the work order ID to supply (max 256 characters)

**Example:**

```
oneqa://application/execute?procedureId=d826477c-fe78-4cc2-9715-
c1e22c578b67&settings.WorkOrderId={"id": "123-456-789"}
```

**Error handling:**

If the supplied id contain more than 256 characters procedure will start but user get an error message in OneQA that the id was not correctly formatted. In this case no id will be added to the running procedure.

#### 2.1.2.1.2 Asset Information

Supplies asset information for the given Asset ID to create or update the information for asset in OneQA.

**setting-type:**

```
settings.AssetInformation =json-value
```

**json-value:**

```
{
    "manufacturer": "Philps",
    "model": "FD2020",
    "serialNumber": "123ABC-def456",
    "firmwareVersion": "version 1.23B",
    "medicalDeviceCategory": "defibrillator",
    "description": "located at department 45C, has long cables",
    "pmDate": "2019-05-15",
    "pmInterval": "P1Y2M10DT2H30M"
    "createUpdateAsset": "false"
}
```

- “manufacturer” string asset manufacturer name (max 256 characters) (optional)
- “model” string asset model name (max 256 characters) (optional)
- “serialNumber” string asset serial number (max 256 characters) (optional)
- “firmwareVersion” string asset firmware version (max 256 characters) (optional)
- “medicalDeviceCategory” string asset medical device category (max 256 characters) (optional)
- “description” string asset description (max ??? characters) (optional)
- “pmDate” DateTime assets last PM date (ISO 8601 date time) (optional)
- “pmInterval” TimeRange assets PM interval (ISO8601 time period) (optional)
- "createUpdateAsset" Boolean if asset should be added/updated, default true (optional)

**Example:**

```
oneqa://application/execute?procedureId=d826477c-fe78-4cc2-9715-
c1e22c578b67&assetId=123-abc &settings.AssetInformation={
    "manufacturer": "Philips",
    "model": "FD2020",
    "serialNumber": "123ABC-def456",
    "firmwareVersion": "version 1.23B",
    "medicalDeviceCategory": "defibrillator",
    "description": "located at department 45C, has long cables",
    "pmDate": "2019-05-15",
    "pmInterval": "P1Y2M10DT2H30M"
```

```
    "createUpdateAsset": "false"
}
```

(the example is pretty formatted for readability, should be all be single line single space)

**Error handling:**

If any of the limits on number of characters is exceeded procedure will start but user get an error message in OneQA that asset information was not correctly formatted. In this case the value in question will be disregarded and left empty if asset is created or not updated if asset exist.

If the “pmDate” or “pmInterval” is not correctly formatted procedure will start but user get an error message in OneQA that asset information was not correctly formatted. In this case the value in question will be disregarded and left empty if asset is created or not updated if asset exist.

If “assetId” is not given as parameter procedure will start but user get an error message in OneQA that asset information without an asset ID. In this case all asset information will be disregarded and no asset created or updated.

#### 2.1.2.1.3 Applied Part Setup

Supplies the applied parts to test for ESA in URI to auto fill in the procedure.

When applied parts information is supplied with the URI this will replace any existing applied parts setup that is included in the procedure.

If the option to “Lock applied parts setup” is checked for the procedure to restrict the user from changing the applied parts setup at runtime they can still be supplied by the URI but the user will not be able to change them.

**setting-type:**

```
settings.AppliedPartSetup=json-value
```

**json-value:**

```
{
    "data": [
        {
            "appliedPartName": "partname",
            "appliedPartType": "B",
            "parts": 1,
            "isAdapter": false,
            "isTestIndividually": false
        },
        ...
    ]
}
```

- “data” struct the applied parts to supply
- “appliedPartName” string the part name (max 256 characters)
- “appliedPartType” enum part type one of: “B”, “BF”, “CF”, “CF-Pad”

- “parts” integer number of parts
- “isAdapter” boolean if adapter is used or not, one of: true, false
- “isTestIndividually” boolean if parts should be tested individual, one of: true, false

**Example:**

```
oneqa://application/execute?procedureId=d826477c-fe78-4cc2-9715-
c1e22c578b67&settings.AppliedPartSetup={
    "data": [
        {
            "appliedPartName": "Part 1",
            "appliedPartType": "B",
            "parts": 1,
        },
        {
            "appliedPartName": "Part 2",
            "appliedPartType": "BF",
            "parts": 2,
        },
        {
            "appliedPartName": "Part 3",
            "appliedPartType": "CF",
            "parts": 3,
            "isAdapter": true,
            "isTestIndividually": true
        }
    ]
}
```

(the example is pretty formatted for readability, should be all be single line single space)

**Error handling:**

The supplied json-value must comply with the general rules related to applied parts, see OneQA documentation for rules related to applied parts. If the supplied json-value do not follow the general structure or do not meet the general applied parts rules the procedure will run but user will get a notification that the applied parts information was incorrect and applied parts data in procedure will not be updated.

## 2.2 Call out from OneQA

Its possible to have OneQA call other applications as result of user actions and state changes in OneQA. The call out can be to just store a result as file on the local computer and/or a call to external application or a REST web API.

### 2.2.1 New Result

When a user is done running a procedure and the result is approved OneQA can store the result and/or make an external call to let some application know a new result is available.

The new result will be shared when it is approved, this will occur at the same time as user is done with procedure or in case the results approval workflow is turned on when the result is approved.

There is two ways to specify how new results should be shared

1. Supply a callback action with externa URI run procedure call in to OneQA
   This will share the result one time only, when that specific result is done or approved
2. Give a default callback action for a procedure
   This will share all results created by that procedure when done or approved

#### 2.2.1.1 Callback in URI

When call in to run a procedure it's possible to include a callback OneQA will do for the new result

```
oneqa://application/execute?procedureId=procedureid
&callbackType=sharedStorage
&callback=callbackuri
```

- callback — The callback URI/URL address to use for new result
  - URL — use http/https URL to call a REST web API
  - URI — use other URI to call local application on the same computer or just store file to local folder
- callbackType — Type of URI call back to do, One of:
  - sharedStorage — Share the JSON result file using SharedStorageAccessManager API
  - tempFile — Share the JSON result file in temp folder on local computer
  - localFolder — Save the JSON result file to folder on local computer

**HTTP callback**

```
oneqa://application/execute?procedureId=procedureid
&callback=http://example.com/some/api/uri?someparameter1=value1&otherparameter=value2
```

If a http/https address is given as callback OneQA will do a POST operation to this address for the new result. The result JSON is include as body to the call. For http/https callback the callbackType parameter is not used and will be ignored. The callback URL can also contain additional parameters as needed.

**URI callback using shared storage**

```
oneqa://application/execute?procedureId=procedureid
&callbackType=sharedStorage
&callback=externalapp://result?someparameter1=value1&otherparameter=value2
```

If a URI is given as callback and shard storage is selected OneQA will call this application using the URI submitted as callback including the result as JSON using the SharedStorageAccessManager API. OneQA will remove the file from SharedStorageAccessManager when application exit.

When the callback is done OneQA will add two parameters to the callback URI

```
externalapp://result?someparameter1=value1&otherparameter=value2
&resultId=d826477c-fe78-4cc2-9715-c1e22c578b67
&resultToken={token}
```

- resultsId — The OneQA GUID unique identifier for the result
- resultsToken — Is used to pick up the file in SharedStorageAccessManager API

The resultsToken is the token returned by SharedStorageAccessManager.AddFile method. UWP application can redeem this token to get access to result file over SharedStorageAccessManager.RedeemTokenForFileAsync

**URI callback using temp file**

```
oneqa://application/execute?procedureId=procedureid
&callbackType=tempFile
&callback=externalapp://result?someparameter1=value1&otherparameter=value2
```

If a URI is given as callback and temp file is selected OneQA will call this application using the URI submitted as callback storing the JSON result in local windows temp folder. OneQA will remove the file from temp folder when application exit.

When the callback is done OneQA will add two parameters to the callback URI

```
externalapp://result?someparameter1=value1&otherparameter=value2
&resultId= d826477c-fe78-4cc2-9715-c1e22c578b67
&resultFile=C:\users\user\appdata\tempfolder
```

- resultsId The OneQA GUID unique identifier for the result
- resultFile Path on the local computer to the JSON results file

**Store result in local folder**

```
oneqa://application/execute?procedureId=procedureid
&callbackType=localFolder
&callback=C:\OneQAresults
```

If a path is given as callback and local folder is selected OneQA will just store the new results JSON file to the local folder given. OneQA will not remove the file from folder, external application will have to do that.

#### 2.2.1.2 Default Callback for Procedure

As and alternative to pass the callback information with the call in its possible to add a default callback when edit a procedure.

The general mechanism is the same as when the callback is included with the call in.

**HTTP callback**

```
https://example.com/api/integration?someAdditionValue=oneqa
```

**URI callback using shared storage**

```
customschema://result?someAdditionValue=oneqa&callbackType=sharedStorage
```

**URI callback using temp file**

```
customschema://result?someAdditionValue=oneqa&callbackType=tempFile
```

**Store result in local folder**

```
C:\tempfolder
```