

Using the SFG20 Facilities-iQ API

**V1.1 17/1/24**

By Mike Talbot

[Overview 2](#_Toc156430858)

[Gaining Access 2](#_Toc156430859)

[Getting an Access Token 2](#_Toc156430860)

[Getting a Sharing Link 2](#_Toc156430861)

[Accessing the API via the GraphQL API 3](#_Toc156430862)

[Accessing the API via the CLI 4](#_Toc156430863)

[The API 5](#_Toc156430864)

[Retrieving Tasks 5](#_Toc156430865)

[Standard Queries 6](#_Toc156430866)

[Tasks 7](#_Toc156430867)

[Schedules 8](#_Toc156430868)

[Frequencies 9](#_Toc156430869)

[Skills 10](#_Toc156430870)

[Example API Output 11](#_Toc156430871)

[Displaying SFG20 Tasks 13](#_Toc156430872)

[Marking a Task as Complete 13](#_Toc156430873)

# Overview

The SFG20 API provides functionality to retrieve data from a customer’s maintenance regimes and to indicate that a task has been completed.

Data can be retrieved in several different formats, depending on your use case. For example, you could retrieve tasks based on the location of the asset being maintained and the frequency of the maintenance required, or you could retrieve tasks based on the schedule to which they belong etc.

You can retrieve data using:

* The GraphQL API available at <https://api.facilities-iq.com/v3.0>, a playground is available for you to try the API
* The SFG20 CLI tool

The tasks retrieved are the abbreviated quick reference step lists for SFG20, not the entire SFG20 schedule. Tasks are abbreviated consistently to enable their display in mobile devices. The step lists are available either as a bullet point block of plain text or as a list of the individual steps in an array, where each entry is a plain text step content.

You should provide the SFG20 Smart Words associated with each task so that a user can type them into the SFG20 Application to access the full schedule contents and all the relevant introductory procedures. If you use the content block, the smart words are appended at the end.

Make sure you read and understand Displaying SFG20 Tasks before implementing a solution using the results of the API.

# Gaining Access

To use the API, you will need an access token provided by SFG20 and a sharing link for the content to retrieve. The sharing link and the access token must both be in date.

## Getting an Access Token

If you have joined the Digital Partner Programme you may contact SFG20 and be issued an access token. When we create you an access token you will specify the domain that identifies you to clients wishing to share data with you. For example, *somecompany.com*.

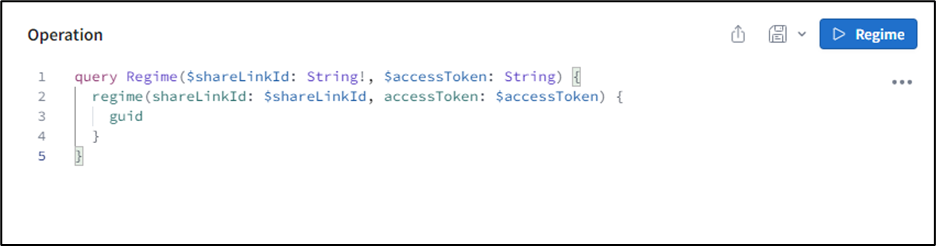
## Getting a Sharing Link

A customer who wishes to share data with you will create a maintenance regime in Facilities-iQ, move it to the “Candidate” or “Live” stages and then create a sharing link. The sharing link should include the domain you specified for your access key as one of the intended recipients. For ongoing integrations you should request a link to a Live regime as Candidate regimes will be deleted if promoted to Live.

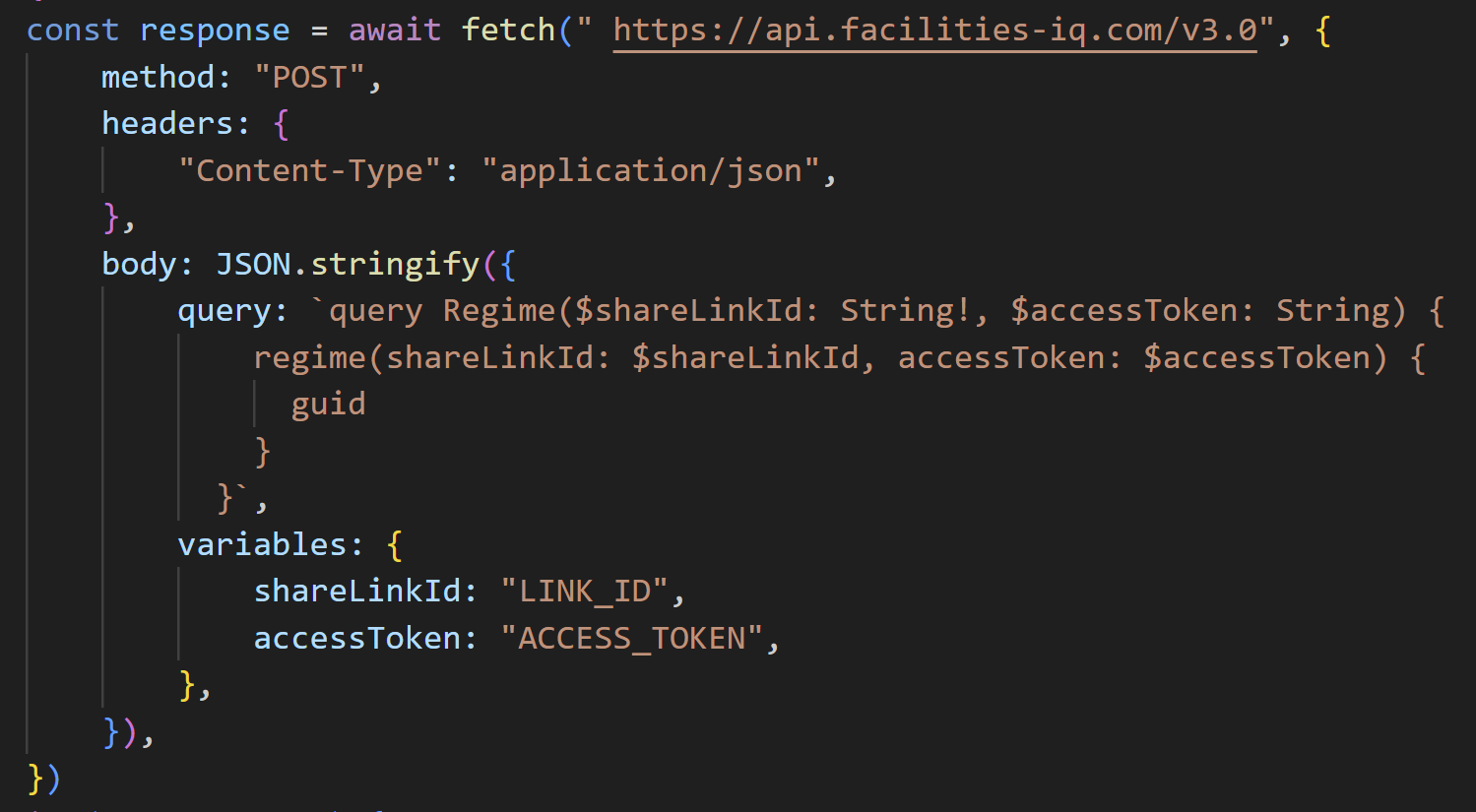
## Accessing the API via the GraphQL API

The GraphQL endpoint for the API is found at <https://api.facilities-iq.com/v3.0>, it includes an IDE available to a browser on that address which allows you to try out queries. When using this tool, you will need to supply both the sharing link key and your access token.

While it is not necessary to install a GraphQL client to interact with the API, the endpoint requires requests to adhere to a GraphQL query structure. This means you can utilize more conventional HTTP request methods, such as JavaScripts ‘fetch’, to communicate with the endpoint, as seen in the following example:

**Query**

**Usage**



Details on the API are found later in this document.

## Accessing the API via the CLI

You can install the CLI on your local computer. Before installing you must have installed NodeJS version 18 or newer.

To install the CLI open a terminal and type:

npm i -g @sfg20/cli

This will install the sfg20 command. Make sure it is installed by typing:

sfg20 --help

You should see the command help being displayed in your terminal.

The CLI comes with 4 built in methods of returning data from the API. *frequencies*, *schedules, skills,* and *tasks*. You can get a list of the current available methods by typing:

sfg20 list

You may use these methods or create your own by writing GraphQL queries and passing them to the CLI.

You retrieve data using the CLI by typing a command, for example to retrieve all of the tasks to the console in XML format you would type the following, replacing SHARELINK and ACCESSTOKEN with the appropriate values:

sfg20 xml SHARELINK ACCESSTOKEN frequencies

If you wanted the data in JSON format, you wanted to store the results in a file called *skills.*json and you wanted the tasks broken down by skill set you would type:

sfg20 json SHARELINK ACCESSTOKEN skills –-file ./skills.json

If you want to retrieve only the changes since a particular date (which includes entries for anything deleted) you can type:

sfg20 xml SHARELINK ACCESSTOKEN frequencies –-since 2023-09-01

You can write your own queries and pass them with the –query option, an examples of this type of query are shown later in this document.

See the –help output for more options.

# The API

## Retrieving Tasks

To retrieve tasks via the API you use the regime query. The parameters are documented in the IDE portal.

|  |  |
| --- | --- |
| Parameter | Purpose |
| shareLinkId:String! | The ID of the share link |
| accessToken:String | The Access Token |
| changesSince:Date | The Date from which changes should be returned |
| forDays:Int = 365 | At present this should be left at 365 to return all values for a year, in future this parameter will be used for additional functionality. |
| minimumInterval:Int | The minimum time between tasks for a task to be returned measured in hours. E.g. 24 will not return tasks that should be performed more than once a day. |
| maximumInterval:Int | The maximum hours between tasks for a task to be returned. E.g. 8760 would mean tasks with an interval more than 1 year were not returned |

The real work of the API comes in the values requested back from the call, by specifying values in the returned object you choose how data should be organised.

The return values are:

|  |  |
| --- | --- |
| Field | Meaning |
| assets | For future use, at present returns entries for the number of assets listed in the regime for the specific schedule. |
| frequencies | Groups results into frequencies of execution – note that overlapping frequencies ARE returned. 3-month frequency includes all tasks that should happen at that visit, e.g. 1 week, 1 month and 3 month tasks. |
| groups | The folder structure of the regime is used to organise tasks, this may frequently represent the physical structure of an organisation such as facilities, spaces, floors, and rooms. |
| schedules | Organise tasks by the schedule that owns them |
| skills | Organise the tasks by the skill required to perform the task |
| tasks | Return the tasks |
| words | For future use, returns the smart words of the regime. |
| guid | Returns a unique ID that represents the current settings of the sharelink. Can be used to validate whether a cache is still relevant. The GUID will change if the settings of the share link have been updated. |

At each of the various levels (e.g. groups or frequencies) you may further choose how to break down the tasks within each value, in this way you can create the shape of data that best fits your purposes.

It’s a good idea to look at the standard queries used by the CLI and use the IDE to experiment.

# Standard Queries

The CLI provides some standard queries that shape the data in useful ways.

|  |  |
| --- | --- |
| Query | Shape |
| tasks | A plain list of all the tasks |
| schedules | A list of the schedules with all their tasks |
| frequencies | A breakdown of the frequencies that tasks should be performed, then all the tasks in each frequency. Frequencies overlap so that at the 6 months mark you would have all 3-month, 1-month, 1-week etc overlapping. |
| skills | First level is the required skill, then for each skill the frequencies of task and then for each frequency, the tasks at that frequency. |

If you want to a make a new query for the API, you can use the IDE and then copy to a file only the breakdown of the results and then reference the file using the –query parameter.

The following pages document the standard queries for the CLI.

## Tasks

task: tasks{  
 ... on APITask {  
 \_\_typename  
 \_status  
 intervalInHours  
 title  
 id  
 url  
 linkId  
 frequency {  
 interval  
 period  
 }  
 skill {  
 Skilling  
 SkillingCode  
 CoreSkillingID  
 }  
 schedule {  
 title  
 code  
 }  
 content  
 classification  
 minutes  
 step: steps  
 }  
 ... on DeletedRecord {  
 \_\_typename  
 id  
 }  
 }

## Schedules

schedules {  
 ... on APISchedule {  
 code  
 id  
 rawTitle  
 title  
 version  
 modified  
 task: tasks {  
  
 ... on APITask {  
 \_\_typename  
 \_status  
 id  
 title  
 content  
 step: steps  
 linkId  
 url  
 }  
 ... on DeletedRecord {  
 \_\_typename  
 id  
 }  
 }  
 }  
 }

## Frequencies

frequencies {  
 ... on APIFrequency {  
  
 frequency {  
 interval  
 period  
 }  
 label  
 tasks {  
 ... on APITask {  
 id  
 content  
 skill {  
 Skilling  
 }  
 steps  
 minutes  
 title  
 where  
  
 classification  
 linkId  
 intervalInHours  
 schedule {  
 title  
 code  
 }  
 \_status  
 }  
 }  
  
 }  
 }

## Skills

skills {  
 ... on APISkill {  
 skill {  
 Skilling  
 SkillingCode  
 }  
 countTasks  
 frequencies {  
 ... on APIFrequency {  
 label  
 frequency {  
 interval  
 period  
 }  
 countTasks  
 tasks {  
 ... on APITask {  
 \_status  
 intervalInHours  
 title  
 id  
 url  
 linkId  
 frequency {  
 interval  
 period  
 }  
 skill {  
 Skilling  
 SkillingCode  
 CoreSkillingID  
 }  
 schedule {  
 title  
 code  
 }  
 content  
 classification  
 minutes  
 steps  
 }  
 ... on DeletedRecord {  
 id  
 }  
  
 }  
  
 }  
 }  
 }  
 }

# Example API Output

Here is an example of the output of the *schedules* standard query: we see one schedule with two tasks. For each task it includes the frequency of the task, the amount of time the task should take, the block of content including the Smart Words and the broken-out individual steps. The output also includes a URL that can be used with a free Facilities-iQ account to view the complete detail of the task and all other introductory procedures. The Smart Words can be typed into the SFG20 application (iOS and Android) to show the complete detail of the task and the introductory procedures.

{

"code": "66-02",

"id": "gpoZQkaq5nsXhA!266",

"rawTitle": "Hydrotherapy Pools",

"title": "Test Regime › Coachmans Cottage › Test - Hydrotherapy Pools",

"version": "4",

"modified": "2022-08-03T11:16:52.2129813Z",

"task": [

{

"\_\_typename": "APITask",

"\_status": null,

"id": "DrH7AyrFUImrkd.gpoZQkaq5nsXhA!266.266.1",

"title": "Hoists - daily checks",

"content": "\* Operate all hoists daily as per the manufacturer's handbook.\n\* Check operation and adjustment of hoist controllers; if found unresponsive or improperly adjusted, initiate repair or adjustment.\n\* Check effectiveness of hoist brakes and clutches; if ineffectiveness is noticed, repair or adjust them immediately.\n\* Visually inspect hoist wire or chain to ensure they are correctly located on drum, sheaves, or pulleys; if incorrectly located, reposition as required.\n\* Visually check hoist supporting structure and all bolted connections to ensure nothing has worked loose; tighten any loose bolts.\n\* Record any hoist defects encountered during daily checks in the daily log book.\n\nSmart Words: quarter minor incident",

"frequency": {

"interval": 1,

"period": "Day"

},

"intervalInHours": 24,

"minutes": 7.5,

"step": [

"Operate all hoists daily as per the manufacturer's handbook.",

"Check operation and adjustment of hoist controllers; if found unresponsive or improperly adjusted, initiate repair or adjustment.",

"Check effectiveness of hoist brakes and clutches; if ineffectiveness is noticed, repair or adjust them immediately.",

"Visually inspect hoist wire or chain to ensure they are correctly located on drum, sheaves, or pulleys; if incorrectly located, reposition as required.",

"Visually check hoist supporting structure and all bolted connections to ensure nothing has worked loose; tighten any loose bolts.",

"Record any hoist defects encountered during daily checks in the daily log book."

],

"linkId": "quarter minor incident",

"url": "http://localhost:3030/sfg20?phrase=quarter+minor+incident`,"

},

{

"\_\_typename": "APITask",

"\_status": null,

"id": "DrH7AyrFUImrkd.gpoZQkaq5nsXhA!266.266.7",

"title": "Log book",

"content": "\* Consult the Log Book for any reported defects and address them as necessary, wearing appropriate personal protective equipment if required.\n\* Record any completed works in the Log Book clearly according to the operating code of practice.\n\nSmart Words: negative fact decorate",

"frequency": {

"interval": 1,

"period": "Month"

},

"intervalInHours": 730,

"minutes": 15,

"step": [

"Consult the Log Book for any reported defects and address them as necessary, wearing appropriate personal protective equipment if required.",

"Record any completed works in the Log Book clearly according to the operating code of practice."

],

"linkId": "negative fact decorate",

"url": "http://localhost:3030/sfg20?phrase=negative+fact+decorate`,"

}

]

},

# Displaying SFG20 Tasks

You may display the relevant information retrieved from the API in your application. The aide-memoire step list is **not** the complete SFG20 task or schedule, all parties should be familiar with the complete schedule and all introductory procedures when undertaking work.

The Smart Words and the task URL provided by the API can be used to freely access the complete schedule in Facilities-iQ or in the SFG20 application. You are therefore required to show one or both task links when you show the content. You may only use the URL in isolation if it is a hyperlink that the user can follow with a single click.

The Smart Words are a good choice for display on mobile devices as they can be used in the SFG20 application.

**It is important to note that a task for the same schedule may differ in different locations within a Maintenance Regime due to tailoring. You should never cache individual schedules by code. The *id* field is unique to each tailorable schedule and task and can be used for caching.**

**You may make Smart Words and step lists available in other forms, such as Excel spreadsheets, Word documents etc. You may not publish, or allow to be published, lists of Smart Words or URLs in a public format, such as on a publicly accessible website. If you include tasks or step lists in a document, they must be accompanied by the relevant Smart Words.**

# Marking a Task as Complete

When tasks are completed, you should call the *completeSharedTask* mutation. You can pass multiple task completions in one call.

The API has provided task ids for each task you retrieved that should be passed to the mutation. For each *taskId* you must supply the date of completion, an optional assetIndex if there are more than one assets for the taskId and an optional JSON metadata object with other information you wish to record (for example the person who completed the task).