STARC API

Oct 24 2020 07:04

Table of contents

No table of contents entries found.

Preamble

Welcome to the world of STARC Interfaces. Here you will find the necessary information for your application to integrate with STARC. Some things will be pretty easy to understand and some might seem too complicated. In the following chapters you will read about the basic requirements, authorization, possibilities of integration, how to get access to the STARC api and best-practices when implementing your integration. As soon as you have any additional questions, feel free to contact us. We will help you as fast as we can.

General information

- Access to the codeBeamer installation is required. Check, if all firewalls are open to port 443.
  - **STARC API**
    - Integration environment: https://starc-backend-for-frontends-int.run-caas.rd.corpintra.net (via intranet) and https://api.starc-int.mercedes-benz.com (via internet)
    - Production environment: https://starc-backend-for-frontends-prod.run-caas.rd.corpintra.net (via intranet) and https://api.starc.mercedes-benz.com (via internet)
  - **STARC GUI**
    - For development purposes: https://starc-dev.rd.corpintra.net (currently only via intranet)
    - Integration environment: https://starc-int.mercedes-benz.com (same URL for access via internet and via intranet)
    - Production environment: https://starc.mercedes-benz.com (same URL for access via internet and via intranet) ***Due to bug in codeBeamer link is not clickable***
- A user with the permission to access the relevant defects/tests AND permission to use the REST API is required
- The base URL for all requests, which will be referenced in the document as <restURI>, is:
  - <environmenturl of api>/starc/v3 (for intranet)
  - <environmenturl of api>/v3 (for internet)
- The REST API uses JSON as exchange format. You will have to make sure, that all relevant special characters will be escaped according to the JSON specification (https://www.freeformatter.com/json-escape.html)
- Detailed information about the codeBeamer API can be found in the codeBeamer wiki (https://codebeamer.com/cb/wiki/5250839)
- Information of how to get access to STARC can be found in the Social Intranet (https://social.intra.corpintra.net/docs/DOC-260726).

Additional documentation

You find additional technical API documentation and how to map the business process to APi calls in the STARC Wiki (STARC Webinars). Check the "Webinar STARC API Training for Daimler Supplier"

Interactive Clients vs. batch systems

Usually when someone talks about a client, he means to start a frontend of an application which talks to a backend. A prominent example at Daimler is the Common Engineering Client (CEC). This frontend includes modules for multiple different backends; the user is able to manipulate the data stored in the backends through the frontend. Having a look at STARC (or codeBeamer) this is also the case: Through a web based frontend it is possible to manipulate data stored in the STARC backend.

From an API point of view, it is a bit different: Every user of the API is a client ... usually called consumer.

In STARC we classify API clients as:

- The application operates in the context of a user. This is usually the case with interactive applications, in which the users actions will be directly translated to API calls.
  => **This is an interactive client**
- The application cannot operate in the context of a user. This is usually the case when data is synchronized between applications in a batch job without any user interaction.
  => **This is a batch system**

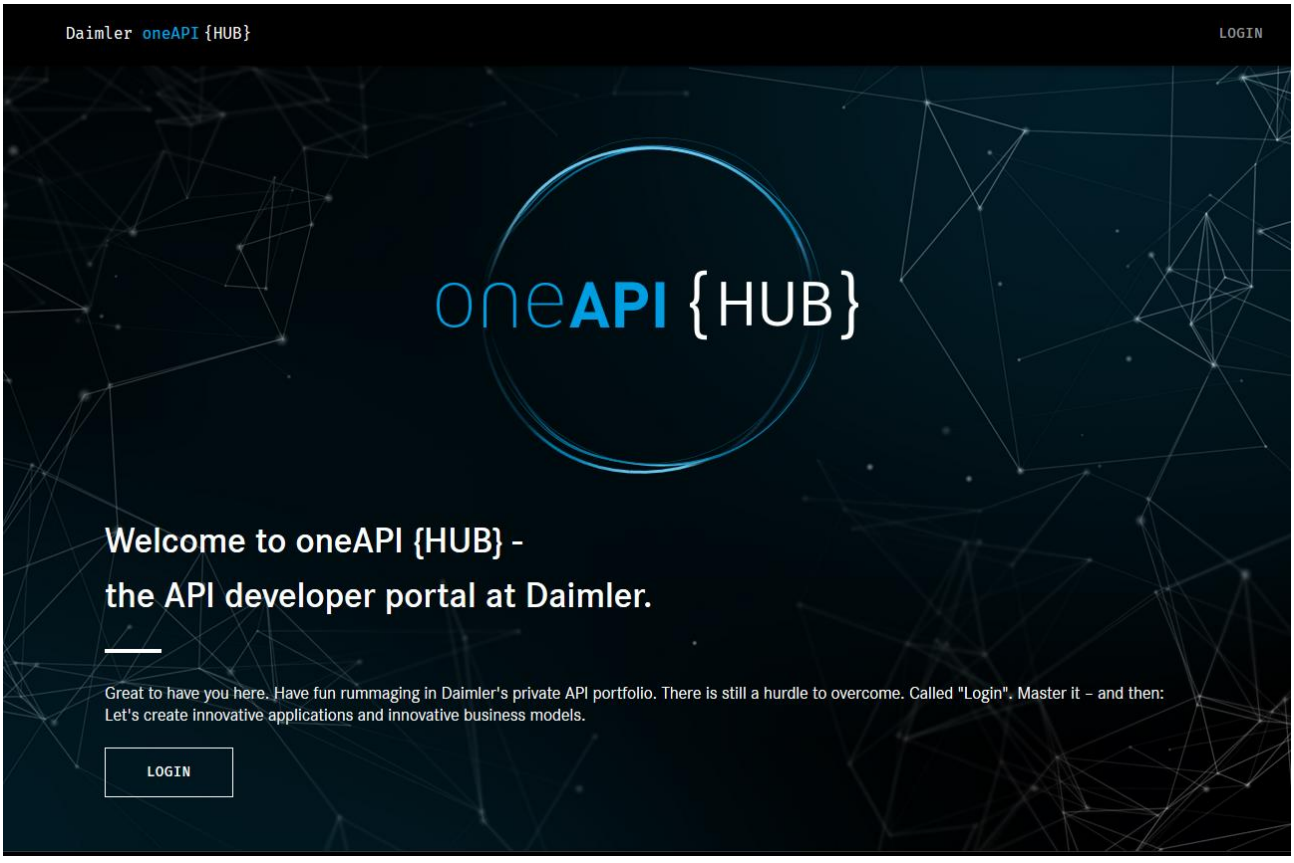
Getting a first impression of the STARC API through the OneAPI developer portal

Essentially the API of codeBeamer is provided as is and the API documentation of codeBeamer itself is also valid as documentation of the STARC API. As the documentation of codeBeamer might be updated before we could update STARC, the current valid API documentation is provided through the oneAPI developer portal. If you do not have access to the OneAPI portal, then you can check https://codebeamer.com/cb/v3/swagger/editor.spr, but be aware, that this might contain changes, which are not in STARC yet.

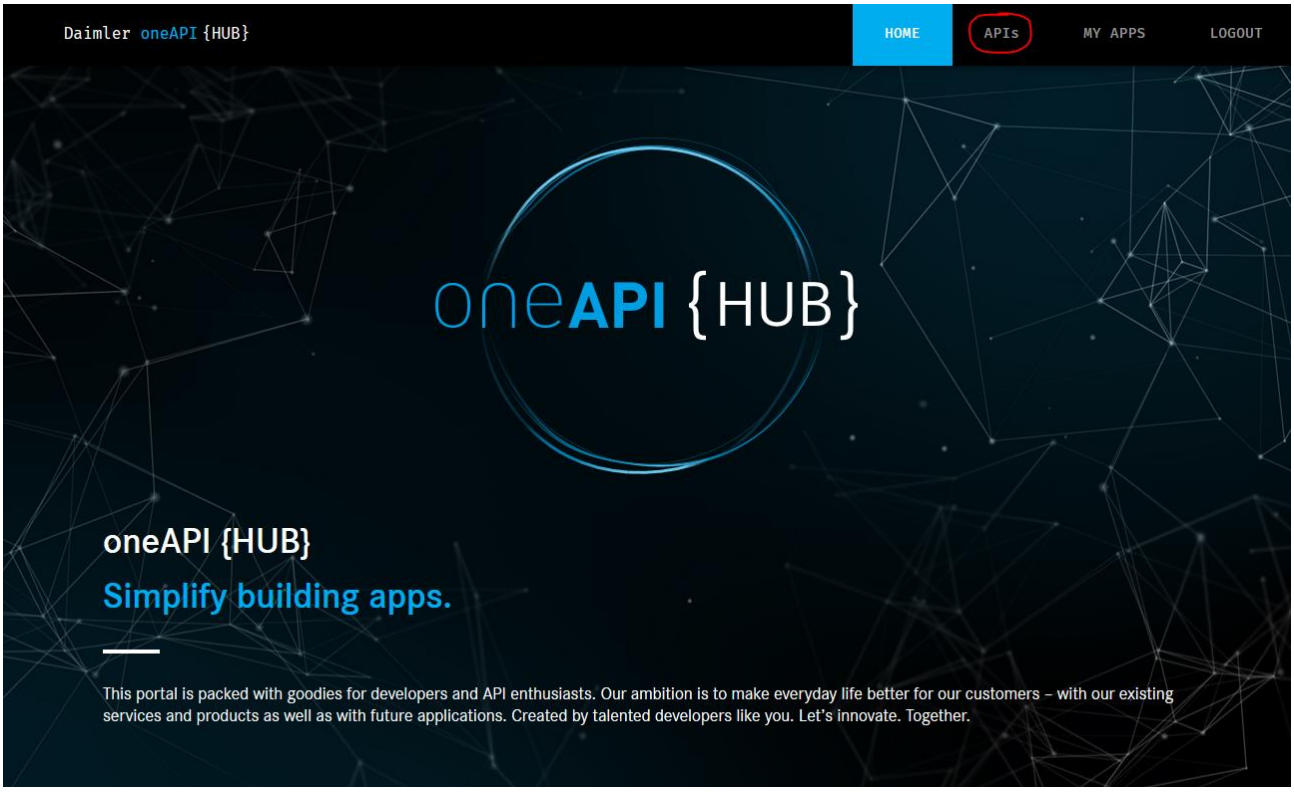
> We just copy the documentation of codeBeamer into the oneAPI portal with only minor changes but without improving the documentation.

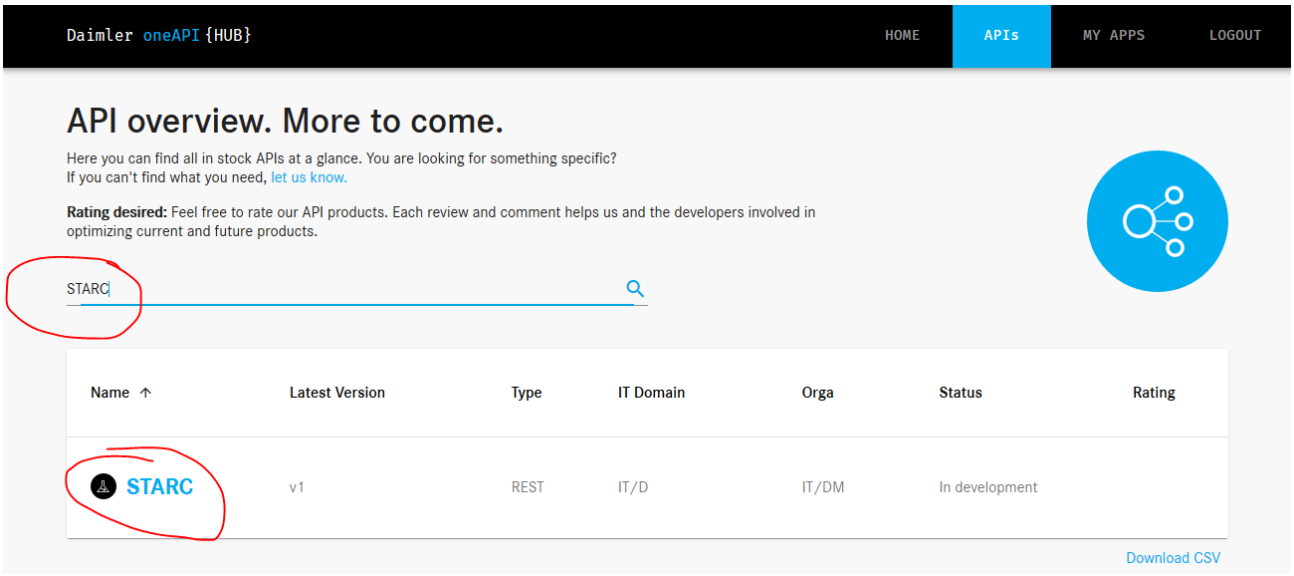
The following steps need to be done, to read the API documentation.

1) Login at the oneAPI developer portal - the oneAPI{HUB}. The hub is accessible here (https://developer.corpinter.net/):

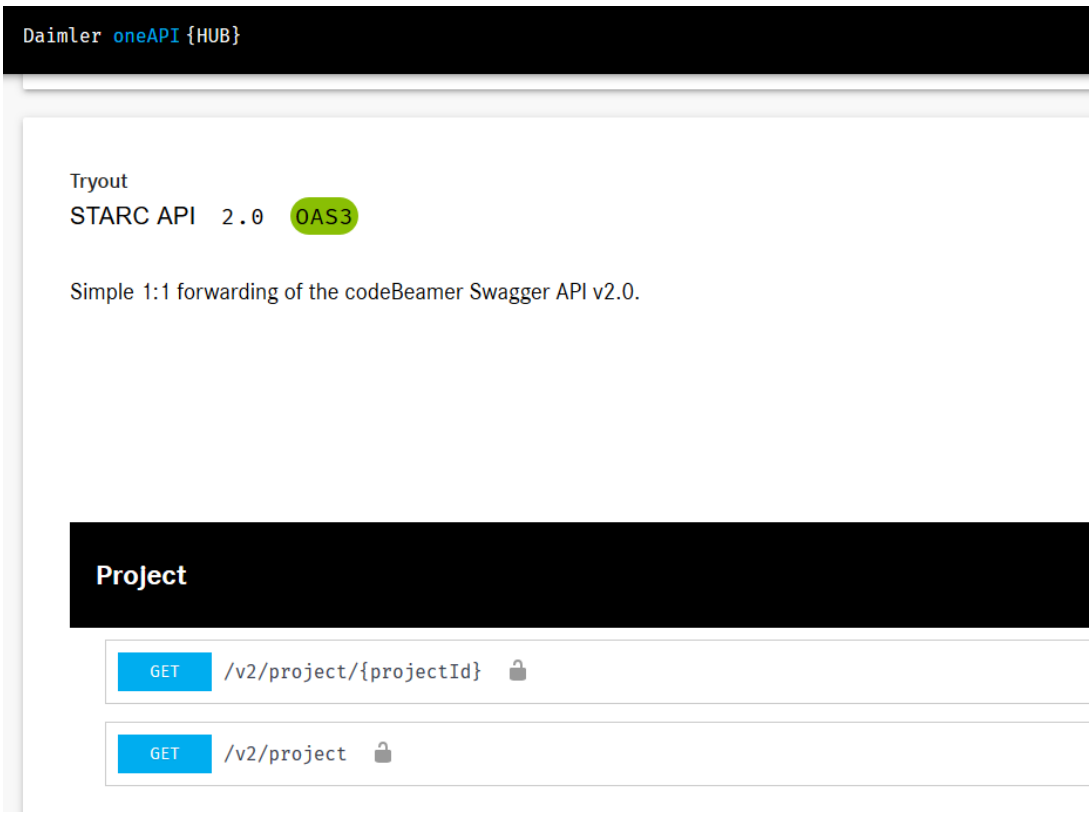


2) Search for the STARC API within the tab "APIs"





3) You can check the documentation after scrolling down.



Using the session manager to access STARC via API

Authorization

The underlying base product 'codeBeamer' uses the new Daimler single-sign-on standard: OpenID Connect. The STARC API is an 1:1 provisioning of the codeBeamer API and thus is supporting OpenID Connect too. If the Session-Manager is used (see below), some requirements of OpenID Connect are already implemented and ready for use.

What is OpenID Connect / OAuth 2.0 and why is that important?

Quick and dirty

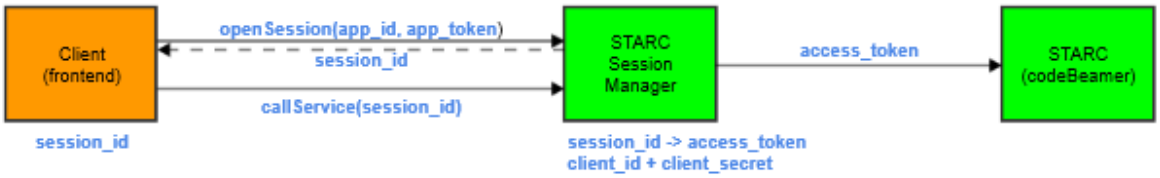
- With OpenID Connect you are able to sign in once and use many applications (single-sign-on)
- OpenID Connect is based on OAuth 2.0
- The STARC api is secured via OpenID Connect. This means, that you need an access token to access the provided services
- To get an access token, you need to integrate with the Daimler authorization servers
- To access the authorization servers, you need a technical user and password - within the context of OAuth 2.0 these are called "client id" and "client secret"

A bit more detail

- The OIDC/OAuth authorization seems to be easy at a first glance, but actually it's more complex than someone might expect.
- To get the required understanding, the following terms and their meaning should be known before starting to integrate with STARC
    - Authorization code flow
    - Implicit flow
    - public client
    - confidential client

- backend for frontend
- Here some links containing some useful documentation
  - Oracle Cloud Primers of the Oracle Learning Library https://www.youtube.com/watch?v=74bP5fwmm8Y&list=PLKCk3OyNwIzuD_jxWu-JddooM2yjX5q99
  - OAuth 2.0 and OpenID Connect (in plain english) https://www.youtube.com/watch?v=996OiexHze0
  - Daimler GAS-OIDC Sharepointhttps://team.sp.wp.corpintra.net/sites/05389/GAS-OIDC/SitePages/DaimlerHome.aspx
    - Notably the "GAS-OIDC Integration Guide"

Integrating with STARC using the STARC Session Manager



Because of requirements in the OpenID Connect protocol, which is implemented by codeBeamer and used to secure the API, it is - at the moment - not possible to use access tokens directly.

As batch systems usually do not operate within the context of a real user, the usage of a technical user is required. To keep it simple, interactive clients and batch systems share the same session handling and will use application IDs and application tokens to authenticate. As users are able to type the login and password on request, the session manager can use the authorization flows specified by OpenID connect. For batch systems, the session manager will map service calls which use the session id, transparently and will allow to read/write data from/to STARC.

To support the recurring user permission revalidation process enforced by ZuLa, the technical user needs to be a PID user. PID users are users which are available within the corporate directory and can be handled as normal users (with some restrictions) ... this includes requesting permissions via ZuLa which will then enforce all processes for granting and revoking permissions.

Authorization

Open session

The authorization for interactive clients and batch systems differ in detail. Both types need to call the authorization endpoint

```
https://starc-backend-for-frontends-int.run-
caas.rd.corpintra.net/auth
```

first.

```
POST /auth
Content-Type: application/json
Accept: */*
Content-Length: 158
Connection: keep-alive
{ "application_id": "<<your id>>", "application_token": "<<your
token>>" }
```

If the application was registered as interactive application, the service will return a redirect to an url, which needs to be opened in a browser window, because it will redirect multiple times including a login window, where the user needs to enter username and password. As last redirect an url which ends in '/confirmed' will be called which marks the end of the authorization flow. The required session id can be extracted from a cookie 'STARC_SESSION' or can be read from the json returned by the 'confirmed' endpoint.

If the application was registered as batch application, the service will return a redirect directly to the confirmation service ('/confirmed') and marks the end of the authorization flow. The required session id can be extracted from a cookie 'STARC_SESSION' or can be read from the json returned by the 'confirmed' endpoint.

Attention:

When redirecting, the session manager makes use of the HTTP return code 303. Make sure your application is able to handle it correctly (https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html).

Close session

To logout, simply call the service

using the HTTP method 'DELETE'. This will close the session passed in the header as 'STARC_SESSION'.

Attention for web clients

When implementing the authorization in web clients, some additional things need to be done:

- In addition to the `application_id` and the `application_token`, you need to pass the property `client_type` with a value of `web`, when calling the `/auth` service. Also you need to provide a `referrer` in the header which is the final target URL to redirect to.
- The result of the `/auth` service is HTTP code 200 and an url in the header `location`. This will prevent the browser to automatically redirect to the Daimler login page and thus gives your application some additional control. The URL in the `location` header needs to be handled by your web client; usually you just change the `window.location` to the provided url.
- After the user provided the login credentials, the browser will be redirected to the URL you provided as `referrer` when calling the `/auth` service.
- As the session cookie is set for the STARC api domain, you will not have access to the session id. If you need to extract the session id just call the `/confirmed` service which will return the session id in a small JSON.
- If you need to check, if your session is still valid, we provide a service `/auth:check`

Example implementation

To get a better check, if and how requests are working for different usecases, we created a java library and an example application to interactively run different usecases.

We provide the source code in the Daimler GitHub:

- https://git.daimler.com/starc/starc-api-client : Client library.
- https://git.daimler.com/starc/starc-api-client-example : Example using the library.

The client library is provided as-is. It was created to have a working example when implementing the integration in the specific application. The library is not ready for production and will not be developed in a way, that you should use it as library in your project. Take the code as an inspiration and do not use it as-is. If you decided to use the library as-is anyway, Daimler cannot be held responsible, if something breaks or is not working as expected.

Where can I find help and ask questions?

STARC Support

Phone: +49 7031 90 – 89500

E-Mail : starc-support@daimler.com