# BILINFO AUTH SERVICES

Third party integration document

Abstract

Bilinfo Auth Services encompass Single-Sign On and OAuth 2.0 mechanisms that must be used when integrating Bilinfo. This document describes the prerequisites and details of integrating into Bilinfo Auth Services.

BilInfo

bilinfo@bilinfo.dk

## Version history

| Version | Date | Authors | Comments |
|---|---|---|---|
| 0.1.0 | 10/09/2018 | Jacob Sønderskov (jsonderskov@ebay.com) | First draft version based on the *Bilinfo integration using the Case Plugin Architecture (2.4.0)* documentation. |
| 1.0.0 | 12/09/2018 | Jacob Sønderskov | First release version. |
| 1.0.1 | 13/09/2018 | Jacob Sønderskov | Change cover and add abstract. |
| 1.1.0 | 17/09/2018 | Jacob Sønderskov | Remove test credentials |
| 1.2.0 | 27/03/2020 | Martin Pløk Svenningsen | Added section specific to CarApi |

## Contents

## A. Disclaimer

Information presented here might be altered by eBay from time to time. Inconsistencies across the document are to be expected and they will be addressed in updates.

Any update will be specified in Version history.

### Data and system integrity

Abuse of the system is forbidden in any regard. If you find a security issue or exploitation outside the original intent of the system, you are expected to report the exploit or bug to the Bilinfo team.

## B. Versioning and Deprecation Policy

Versioning in Bilinfo Services is essential to achieving our vision behind Partner integrations in Bilinfo. Using the versioning principles described below will allow for your Bilinfo integrations to remain stable and fully functional as the Bilinfo business continues to evolve and mature.

### New Versions of the Bilinfo Services

The versioning principles employed in Bilinfo Services largely follow that of the *Semantic Versioning Specification*[1]. The Semantic Versioning Specification, in short, specifies a version increment based on the backwards compatibility of the API or Web Service. A summary of the specification can be seen in the following Listing B-1:

> Given a version number `MAJOR.MINOR.PATCH`, increment the:
>
> 1. `MAJOR` version when you make incompatible API changes,
> 2. `MINOR` version when you add functionality in a backwards-compatible manner, and
> 3. `PATCH` version when you make backwards-compatible bug fixes.
>
> Additional labels for pre-release and build metadata are available as extensions to the `MAJOR.MINOR.PATCH` format.

*Listing B-1: Semantic Versioning Specification 2.0.0 summary*

The types of changes that are minor version changes and backward compatible are:

- Adding a new method (`GET`, `POST` etc.) to an API
- Adding a new property to the method response payload
- Adding a new *non-personal data*[2] property to an `iframe` communication

The types of changes that are major version changes and not backward compatible are:

- Removing existing method (`GET`, `POST` etc.) from an API
- Renaming existing method path
- Changing request body or query string for existing method
- Changing method response structure and/or property names
- Removing a property from an `iframe` communication
- Renaming a property from an `iframe` communication
- Renaming a message in an `iframe` communication
- Adding a new *personal data*[2] property to an `iframe` communication

[1] https://semver.org

[2] Personal data as defined in Regulation (EU) 2016/679 of 27 April 2016 (GDPR) and the Danish Data Protection Act.

In general, new major versions of Bilinfo Services will only be introduced, when existing interfaces does not allow for further evolution and improving of our Partner integrations without modification. Due to the need for Partner action, major versions are used as a last resort and are as such very rare. Minor version updates will require no Partner action.

### Updating your Bilinfo integration

Updating your Bilinfo integration to support a new major version is non-optional as the existing integration paradigm is fundamentally changed. It is as such not possible to opt out without risking major problems with your Bilinfo integration. Minor versions, however, are fully optional, but may contain new fields, which may enrich the experience and value of your Bilinfo integration.

To assist Partners in upgrading their Bilinfo integration with minimal efforts, each major version will be associated with a *migration chapters* added to this document. Minor version changes are specified primarily in the Version History and is subject to the reader to adhere to the changes.

### Deprecation Policy and Supported Versions

Bilinfo Services will support older versions for a grace period appropriate to the contractual obligations. After that time, integrations based on older versions may no longer work or experience severe operational issues.

# 1. Introduction

## Purpose and Scope

This document will describe the process of authenticating and authorizing within the Bilinfo.net context. All Bilinfo services and data streams are protected by either Authentication and/or Authorization schemes. As such, any consumer must adhere to the technical specifications found in this document.

## 1.1 References

Documents relevant to the reading of this document are listed here. Links and other external resources accessible via the internet are referenced via footnotes relative to the term or technology. You should have access to every document mentioned in this list. If that is not the case, contact Bilinfo.
1.2

*Table 1-1: Document references*

| **Document name** | **Description** | **Link** |
|---|---|---|
| Bilinfo Case Plugin (Integration) | Describes the integration details for a Case Plugin integration into Bilinfo.net | TBA |
| Bilinfo Shared Services (Integration) | Describes a number of services made available to integrating party, enabling access to additional information to e.g. Dealers. | TBA |

## 1.3 Definitions and acronyms

The definitions and acronyms defined in Table 1-2 cover frequently used concepts, terms and acronyms used throughout this document. It is suggested that the reader acquaints him- or herself with the key concepts and refer to this list, when in doubt.

*Table 1-2: Definitions and acronyms*

| **Term/acronym** | **Definition** |
|---|---|
| Single-Sign On | Mechanism for sharing of a single identity provided by an identity provider for the means of authenticating towards one or more applications. |
| SSO | Abbreviation for Single-Sign On |
| Bilinfo Services | Includes – but not limited to – Bilinfo Auth Services, Bilinfo Shared Services, Bilinfo Finance Offer On Platform integrations and Bilinfo Case Plugin integrations. |
| Partner | Synonym for the integrating party. |
| Plugin | Short for Case Plugin found in the Bilinfo Case Plugin (Integration |

# 2. Prerequisites

One or more of the following prerequisites must be complied to before attempting to contact the Bilinfo Authentication Service and/or Bilinfo Authorization Service.

## 2.1 Bilinfo Authentication Service registration

*This step is only necessary if you're building a* Bilinfo Case Plugin (Integration *into Bilinfo.net*.

In order to use the Single Sign-On (SSO) infrastructure in Bilinfo.net, the Plugin must be registered with the Bilinfo Authentication Service. The registration process is manual and requires you to contact Bilinfo.

The information you must provide and request can be found in Table 2-1 and must be used in the integration specified in the Bilinfo Authentication Service chapter.

*Table 2-1: Bilinfo Authentication Service registration*

| Name | Description | Supplied by |
|---|---|---|
| `redirect_url` | A redirect URI used in the | Integrating party |
| `client_id` | An identifier representing the client (a.k.a. username) | Bilinfo |
| `client_secret` | A secret string (a.k.a. password) | Bilinfo |

## 2.2 Bilinfo Authorization Service registration

*This step is mandatory in any integration with Bilinfo.net*.

All Bilinfo API services are protected by authorization mechanism, which requires the integrating party must be registered with the Bilinfo Authorization Service. The registration process is manual and requires you to contact Bilinfo.

The information you must request can be found in Table 2-2 and must be used in the integration specified in the Bilinfo Authorization Service chapter.

*Table 2-2: Bilinfo Authorization Service registration*

| Name | Description | Supplied by |
|---|---|---|
| `client_id` | An identifier representing the application | Bilinfo |
| `client_secret` | A secret string (a.k.a. password) | Bilinfo |
| `scope` | One or more scopes relative to the consumed services, e.g. found in the Bilinfo Shared Services or Bilinfo Case Plugin (Integration | Partner |

# 3. Bilinfo Authentication Service

This chapter describes the Bilinfo Authentication Service and the OpenID Connect protocol available for use in Case Plugin integration in further detail. The reader should refer to this, if further understanding of the authentication process is required.

The exact workings of the OpenID Connect protocol is outside the scope of this appendix, but information pertaining to this may be found at the OpenID Connect site's specification section[3]. Developers are encouraged to familiarize themselves with the protocol specification.

The Bilinfo Authentication Service supports two OpenID Connect flows: Authorization Code and Implicit flow.

The following sections will provide detailed insight for these flows in regards to the Bilinfo Authentication Service and Single Sign-On (SSO).

## 3.1 Authorization Code

This section will cover the specifics of the Authorization Code relative to the Plugin context. In this flow, the tokens (`id_token` and `access_token`) are obtained Server-Side. Table 3-1 describes the interaction in detail.

*Table 3-1: Authorization Code flow*

| | Step | Description | Location |
|---|---|---|---|
| 0 | The Host loads the Plugin | The Plugin is first loaded due to user action | Plugin webserver |
| 1 | Client prepares an Authentication Request containing the desired request parameters | The Plugin webserver prepares a redirect url and instructs the user agent to navigate to it (302) | Plugin webserver |
| 2 | Client sends the request to the Authorization Server | The user agent navigates to the authorization endpoint | User agent |
| 3 | Authorization Server Authenticates the End-User | If the user is not authenticated, he is prompted to log in | Bilinfo Authentication Service |
| 4 | Authorization Server obtains End-User Consent/Authorization | Permission is automatically granted for the allowed scopes | Bilinfo Identity Server |
| 5 | Authorization Server sends the End-User back to the Client with an Authorization Code | The user agent is instructed to navigate to the `redirect_uri` specified in step 1. A query string, `code`, is also appended and it is meant to be used to obtain tokens | 1. Bilinfo Identity Server<br>2. User Agent<br>3. Plugin webserver |
| 6 | Client requests a response using the Authorization Code at the Token Endpoint | Using the code obtained in step 5, tokens are requested server side | Plugin webserver |
| 7 | Client receives a response that contains an ID Token and Access Token in the response body | `id_token` and `access_token` are now retrieved | Plugin webserver |

[3] http://openid.net/specs/openid-connect-core-1_0.html

|  |  |  |  |
|---|---|---|---|
| 8 | Client validates the ID token and retrieves the End-User's Subject Identifier | The token needs to be validated and information extracted out of it.<br>At this point the user is authenticated in the Plugin system | Plugin webserver |
| 9 | Client is allowed to access the protected resource | The user is authenticated, respond with the Plugin user interface | 1. Plugin webserver<br>2. User Agent |

### 3.1.1 Endpoints

The following endpoints must be used when authenticating a user through the Authorization Code.

|  |  |
|---|---|
| • Authorization endpoint | https://www.bilinfo.net/oauth/connect/authorize |
| • Token endpoint | https://www.bilinfo.net/oauth/connect/token |
| • Full configuration | https://www.bilinfo.net/oauth/.well-known/openid-configuration |

Information regarding Token validation may be found at http://openid.net/specs/openid-connect-core-1_0.html#IDTokenValidation.

#### *Parameters*

Table 3-2 shows the parameters, which must be passed to the Authorization endpoint as query strings.

*Table 3-2: Authorization Code flow parameters*

| Name | Description |
|---|---|
| `client_id` | Plugin specific client identifier |
| `response_type` | code |
| `redirect_uri` | One of the preconfigured `redirect_uri`, an unknown uri will cause the login to fail |
| `scope` | openid profile |
| `state` | RECOMMENDED. Opaque value used to maintain state between the request and the callback. Typically, Cross-Site Request Forgery (CSRF, XSRF) mitigation is done by cryptographically binding the value of this parameter with a browser cookie. |

### 3.1.2 Examples

The following example describes how the Authorization Code authentication redirect looks (Listing 3-1), what parameters are required (Table 3-2), how the code is retrieved through the redirect (Listing 3-2) and how a token is retrieved using the `code` and `client_secret`.

#### *Authentication redirect*

Listing 3-1 shows how the initial redirect is performed by the Plugin Web Server to the Bilinfo Authentication Service Authorization endpoint specifying that it should use the Authorization Code.

```
https://www.bilinfo.net/oauth/connect/authorize
  ?client_id=testclient_authcode
  &response_type=code
  &redirect_uri=http://bin.sso.redirect/
  &scope=openid%20profile
  &state=cba56666-4b12-456a-8407-3d3023fa1002
```

*Listing 3-1: Authorization Code authentication redirect example*

#### *Code retrieval*

Following the successful authentication, the Bilinfo Authentication Service will redirect back to the `redirectUrl` passing along the code to the Plugin Web Server.

```
http://bin.sso.redirect/
  ?code=82061f0f0a83e0ee574a79fc51672850
  &state=cba56666-4b12-456a-8407-3d3023fa1002
  &session_state=B8cBk7WqGoN9uje-JO5J0XG6q8lRrjbmdCp1HvIv4-
A.b63285ee29c9fa9932d15fa6cfe0b3ee
```

*Listing 3-2: Authorization Code redirect code retrieval example*

#### *Token retrieval*

Afterwards, the Plugin Web Server can exchange the `code` and his `client_secret` for access tokens (this happens server-side). The Client sends the parameters to the Token endpoint using the HTTP POST method and the Form Serialization. An example of this request can be seen in Listing 3-3 with the associated response seen in Listing 3-4.

```
POST https://www.bilinfo.net/oauth/connect/token
Content-Type: application/x-www-form-urlencoded

client_id=testclient_authcode&client_secret=N4heDUbtKh3K9VivL1Eh919vbk4RldME&grant_
type=authorization_code&code=82061f0f0a83e0ee574a79fc51672850&redirect_uri=http://b
in.sso.redirect/
```

*Listing 3-3: Authorization Code token retrieval request example*

```
{
  "id_token": "
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IjFiMTE3cDZ6UUZsQzZUbFNSS1IxcUZmSDBxTSI
sImtpZCI6IjFiMTE3cDZ6UUZsQzZUbFNSS1IxcUZmSDBxTSJ9.eyJpc3MiOiJodHRwczovL2JpbGluZm8ub
mV0IiwiYXVkIjoidGVzdGNsaWVudF9hdXRoY29kZSIsImV4cCI6MTQ4NDU3MjU3MCwibmJmIjoxNDg0NTcy
MjEwLCJpYXQiOjE0ODQ1NzIyMDksInNpZCI6ImZiNDhiOGNmMWZhOGFiZWNmNTM2YjZjZDYxYjcyOGZlIiw
ic3ViIjoiY2xhemFyQGViYXkuY29tIiwiYXV0aF90aW1lIjoxNDg0NTYwODAwLCJpZHAiOiJpZHNydiIsIm
5hbWUiOiJjbGF6YXJAZWJheS5jb20iLCJ1c2VySWQiOiIxMDQyMCIsImFtciI6WyJwYXNzd29yZCJdfQ.AP
8i0i6Yq7V6fDCOdofHv_9RgGV1Xbq4zQ7_gK71v98UJ0LE__4UdSg_y95rbug5nrigiSVJtpb8bsXwLHOJ_
99UEkRvDo6DrymCKc0Mynq7dsW3ZVITm-
P9JiZkGNJBqdG9cEyJH4WkyC1zxoldrEBjUouy6WeFd7979Q2BgYE5P4vWx1-
CRCgrD7AniT63gSkz6ONWHZTrcXBQXo5V8JjH6LOZdqEK1mDF2wxG6A7np1JwpgL1VVpTSdWvkThAMpn7sU
QnODRh56TdiydYgX6zfjmgylHjxSUwjqhyCw2GFGfUnlOLnL6SOhp0V8I2ZrbunQvFUepGvzAW6XkqBA",
  "access_token": "9f92f17660cd0509c2ced6fe7d2d6a8e",
  "expires_in": 360,
  "token_type": "Bearer"
}
```

*Listing 3-4: Authorization Code token retrieval response example*

Note that if you are trying to reuse the same `code` several times, you will get an `invalid_grant` error. You will also receive an error if you try to specify a `redirect_uri` other than the one you specified when obtaining the `code`.

## 3.2 Implicit

This section will cover the specifics of the Implicit relative to the Plugin context. In this flow the tokens (`id_token` and `access_token`) are obtained client-side via the user agent. Table 3-3 describes the interaction in detail. Please note that URI fragments are client-side specific, hence they will not be available server-side at Step 5; it is only in Step 6 that they will become available.

*Table 3-3: Implicit flow*

|  | Step | Description | Location |
|---|---|---|---|
| 0 | The Host loads the Plugin | The Plugin is first loaded due to user action | Plugin webserver |
| 1 | Client prepares an Authentication Request containing the desired request parameters | The Plugin webserver prepares a redirect url and instructs the user agent to navigate to it (302) | Plugin webserver |
| 2 | Client sends the request to the Authorization Server | The user agent navigates to the authorization endpoint | User agent |
| 3 | Authorization Server Authenticates the End-User | If the user is not authenticated, he is prompted to log in | Bilinfo Identity Server |
| 4 | Authorization Server obtains End-User Consent/Authorization | Permission is automatically granted for the allowed scopes | Bilinfo Identity Server |
| 5 | Authorization Server sends the End-User back to the Client with an ID Token and, if requested, an Access Token | The user agent is instructed to navigate to the `redirect_uri` specified in step 1. The `redirect_uri` is enriched with a Fragment containing the tokens. | 1. Bilinfo Identity Server<br>2. User Agent<br>3. Plugin webserver<br>4. Plugin client-side |
| 6 | Client validates the ID token and retrieves the End-User's Subject Identifier | The tokens are now accessible client-side, you can chose to validate and authenticate the user there or delegate this to the server-side. | Plugin client-side |

### 3.2.1 Endpoints

The following endpoints must be used when authenticating a user through the ImplicitAuthorization Code.

- Authorization endpoint https://www.bilinfo.net/oauth/connect/authorize
- Full configuration https://www.bilinfo.net/oauth/.well-known/openid-configuration

Information regarding Token validation may be found at http://openid.net/specs/openid-connect-core-1_0.html#ImplicitIDTValidation

#### *Parameters*

Table 3-4 shows the parameters, which must be passed to the Authorization endpoint as query strings.

*Table 3-4: Implicit flow redirect parameters*

| Name | Description |
|---|---|
| `client_id` | Partner specific client identifier |
| `response_type` | token id_token |
| `redirect_uri` | One of the preconfigured `redirect_uri`, an unknown uri will cause the login to fail |
| `scope` | openid profile |
| `nonce` | To mitigate replay attacks, a nonce value must be included to associate a client session with an id_token. The client must generate a random value associated with the current session and pass this along with the request. This nonce value will be returned with the id_token and must be verified to be the same as the value provided in the initial request. |

### 3.2.2 Examples

The following example describes how the Implicit authentication redirect looks (seen in Listing 3-5), what parameters are required (seen in Table 3-4) and how the token is retrieved (implicitly) on the redirect URL (seen in Listing 3-6).

#### *Authentication redirect*

Listing 3-5 shows how the initial redirect is performed by the Plugin Web Server to the Bilinfo Authentication Service Authorization endpoint specifying that it should use the Implicit.

```
https://www.bilinfo.net/oauth/connect/authorize
  ?client_id=testclient_implicit
  &response_type=token%20id_token
  &redirect_uri=http://bin.sso.redirect/
  &scope=openid%20profile
  &nonce=cba56666-4b12-456a-8407-3d3023fa1002
```

*Listing 3-5: Implicit flow authentication redirect example*

### *Token retrieval*

Once the authentication process is complete, the user agent is instructed to navigate to the specified `redirect_uri` to which the `id_token` is appended as a fragment fragment. An example of this redirect can be seen in Listing 3-6.

```
http://bin.sso.redirect/#id_token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IjFiM
TE3cDZ6UUZsQzZUbFNSS1IxcUZmSDBxTSIsImtpZCI6IjFiMTE3cDZ6UUZsQzZUbFNSS1IxcUZmSDBxTSJ9
.eyJpc3MiOiJodHRwczovL2JpbGluZm8ubmV0IiwiYXVkIjoidGVzdGNsaWVudF9pbXBsaWNpdCIsImV4cC
I6MTQ4NDU3MjcyNiwibmJmIjoxNDg0NTcyMzY2LCJub25jZSI6ImNiYTU2NjY2LTRiMTItNDU2YS04NDA3L
TNkMzAyM2ZhMTAwMiIsImlhdCI6MTQ4NDU3MjM2NiwiYXRfaGFzaCI6InRabXlkZUFfMVdjd1VyOHdWeEhW
T0EiLCJzaWQiOiJmYjQ4YjhjZjFmYThhYmVjZjUzNmI2Y2Q2MWI3MjhmZSIsInN1YiI6ImNsYXphckBlYmF
5LmNvbSIsImF1dGhfdGltZSI6MTQ4NDU2MDgwMCwiaWRwIjoiaWRzcnYiLCJuYW1lIjoiY2xhemFyQGViYX
kuY29tIiwidXNlcklkIjoiMTA0MjAiLCJhbXIiOlsicGFzc3dvcmQiXX0.aUeuK5qc8AO831hKPFZW0USXf
mGPa5X11_SalP_4Lcuo4tzLV2XfqUKKItTJ-GiptFBWaBDfQ4VjEahodUyGr7evjeipem3Fuso-
NwQJQUgxiGovI1-Omz8CMv_0bYsqmheC-QWT-
0HksMrw8cJhFDK8QZCpkKAIBGzAZi9kSeyrA_PMniWZ3qqFUMLhVH6GdRDKwZTq1CTG45GoQvhA46aOMLNs
o-8lBZ6ndioLXdWRq2tP_wmwhi01z56soAKBYEDi1R0AXxhG8o08R1O-
kbATKJgy7uNufNUXOqo3TEf6zTAyEjli_72hxorPe6z2FzeVJ7wNPpGZ1ugABrLxyg&access_token=c32
128da6c87cb7ea2bab68709116638&token_type=Bearer&expires_in=360&scope=openid%20profi
le&session_state=VQ9IoLEV-l-
f6SgDqc0OYkhZbb9oM34rRWYLuX_ulOI.47a98bd5b7cf5f79db55795b75c79a37
```

*Listing 3-6: Implicit flow token retrieval example*

Once the response of the above request reaches the client-side, the tokens can be extracted from the fragment.

# 4. Bilinfo Authorization Service

This chapter describes the Bilinfo Authorization service and OAuth 2.0 protocol, which is used when communicating with Bilinfo Shared Services and other Bilinfo Services.

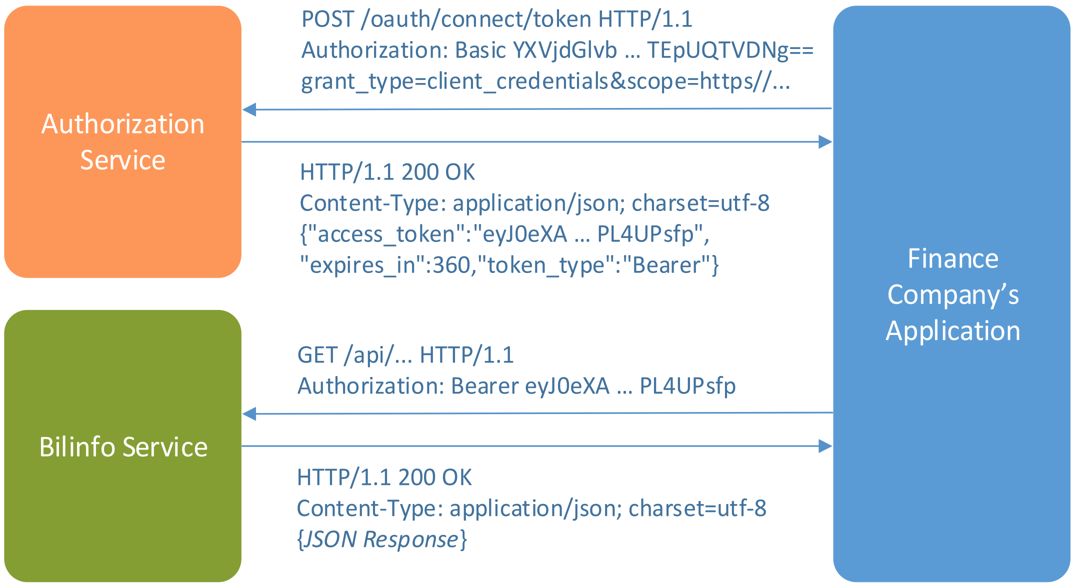
The OAuth 2.0 protocol is simpler than the OpenID Connect in that no redirection occurs, but you simply request an access token, add it to your request header and the endpoint will then validate whether or not you have access to the given information, with that particular token.

The following sections will cover the base interaction, that the Client (e.g. Plugin) must implement when requesting information from Bilinfo.

## 4.1 Client credentials flow

OAuth 2.0 is used for machine-to-machine communication where the application acts on its own behalf and not on behalf of a particular user. The flow allows exchanging a `client_id`, `client_secret` and a `scope` for an `access_token`. The `access_token` can then be used to call the desired API.

An example of the authorization process – using the Client credentials flow – can be seen in Figure 4-1. This example is covered in further detail in the Example section.



```
POST /oauth/connect/token HTTP/1.1
Authorization: Basic YXVjdGlvb ... TEpUQTVDNg==
grant_type=client_credentials&scope=https//...
```

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
{"access_token":"eyJ0eXA ... PL4UPsfp",
"expires_in":360,"token_type":"Bearer"}
```

```
GET /api/... HTTP/1.1
Authorization: Bearer eyJ0eXA ... PL4UPsfp
```

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
{JSON Response}
```

*Figure 4-1: Authorization process*

### 4.1.1 Endpoints

Table 4-1 shows the available endpoints in QA and Production.

*Table 4-1: Bilinfo Authorization Service endpoints*

| Environment | Endpoint |
|---|---|
| QA | `https://qa01-accounts.ecgh.dk/oauth/connect/token` |
| Production | `https://accounts.ecgh.dk/oauth/connect/token` |

*Parameters*

The parameters that must be provided in the requests can be seen in Table 4-2.

*Table 4-2: Client Credentials flow parameters*

| Location | Name | Description |
|---|---|---|
| header | `Authorization` | Basic authentication containing `client_id:client_secret` base64 encoded |
| path | `grant_type` | `client_credentials` always |
|  | `scope` | The scope for which an `access_token` is desired. See the respective services description for this. |

### 4.1.2 Examples

This example simulates the steps required for calling an API residing at https://fictios.api.com/getsomething. For authentication, the credentials found in Table 4-3 are used with a *base64 encoded* as required by the Bilinfo Authorization Service.

*Table 4-3: Fictitious example credentials*

| | |
|---|---|
| `client_id` | myclientid |
| `client_secret` | mysecret |

For authorization, the following scope is required https://scope.required.by.api/, and will be aimed at the QA environment Bilinfo Authorization Service.

Examples of the HTTP communication performed from token acquisition to API data acquisition can be seen in Listing 4-1, Listing 4-2 and Listing 4-3, respectively.

### *Token request*

A token request to the QA environment can be seen in Listing 4-1.

```
POST /oauth/connect/token HTTP/1.1
Host: qa1-accounts.ecgh.dk
Authorization: Basic bXljbGllbnRpZDpteXNlY3JldA==
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&scope=https%3A%2F%2Fscope.required.by.api%2F
```

*Listing 4-1: Bilinfo Authorization Service token request*

### *Token response*

The response from the Listing 4-1 token request can be seen in Listing 4-2.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8

{
      "access_token":   "eyJ0eXA…PL4UPsfp",
      "expires_in":     360,
      "token_type":     "Bearer"
}
```

*Listing 4-2: Bilinfo Authorization Service token response*

### *Access protected endpoint*

Having obtained an access_token, we can now use it to call the API as seen in Listing 4-3.

```
GET /getsomething HTTP/1.1
Host: fictios.api.com
Authorization: Bearer eyJ0eXA…PL4UPsfp
```

*Listing 4-3: Fictitious API request with Bilinfo Authorization Service Bearer token*

# 5. CarApi specific integration

This chapter describes the Bilinfo Authentication Service and the OpenID Connect protocol available for use in CarApi integration in further detail. The reader should refer to this, if further understanding of the authentication process is required.

The exact workings of the OpenID Connect protocol is outside the scope of this appendix, but information pertaining to this may be found at the OpenID Connect site's specification section[4]. Developers are encouraged to familiarize themselves with the protocol specification.

CarApi supports one OpenID Connect flow: Authorization Code.

The following sections will provide detailed insight for these flows in regards to the Bilinfo Authentication Service and Single Sign-On (SSO).

## 5.1 Authorization code

### *Parameters*

Table 3-21 shows the parameters, which must be passed to the Authorization endpoint as query strings.

*Table 5-1: Authorization Code flow parameters*

| Name | Description |
|---|---|
| `client_id` | CarApi specific client identifier |
| `response_type` | code |
| `redirect_uri` | One of the preconfigured `redirect_uri`, an unknown uri will cause the login to fail |
| `scope` | openid profile dealer https://carapi.ecgh.dk/auth/write https://carapi.ecgh.dk/auth/read offline_access<br>Note: Only specify write when you actually need it, else omit it from the list of scopes. Also, offline_access should only be specified when requesting a refresh token. |

[4] http://openid.net/specs/openid-connect-core-1_0.html

### 5.1.1 Examples

The following example describes how the Authorization Code authentication redirect looks (Listing 3-1), what parameters are required (Table 3-21), how the code is retrieved through the redirect (Listing 3-2) and how a token is retrieved using the `code` and `client_secret`.

#### *Authentication redirect*

Listing 3-1 shows how the initial redirect is performed by the Bilinfo Authentication Service Authorization endpoint specifying that it should use the Authorization Code.

```
https://www.bilinfo.net/oauth/connect/authorize
?client_id=yourclientid
&response_type=code
&redirect_uri=yourredirecturi
&scope=openid%20profile%20dealer%20https%3A%2F%2Fcarapi.ecgh.dk%2Fauth%2Fwrite%20ht
tps%3A%2F%2Fcarapi.ecgh.dk%2Fauth%2Fread
```

*Listing 5-1: Authorization Code authentication redirect example*

#### *Code retrieval*

Following the successful authentication the Bilinfo Authentication Service will redirect back to the `redirectUrl` passing along the code.

#### *Token retrieval*

Afterwards, you can exchange the `code` and the `client_secret` for access tokens. Send the parameters to the Token endpoint using the HTTP POST method and the Form Serialization. An example of this request can be seen in Listing 3-33 with the associated response seen in Listing 3-44.

```
POST https://www.bilinfo.net/oauth/connect/token
Content-Type: application/x-www-form-urlencoded

code=codefrompreviousstep&grant_type=authorization_code&client_id=yourclientid&clie
nt_secret=yourclientsecret&redirect_uri=yourredirecturi
```

*Listing 5-2: Authorization Code token retrieval request example*

```
{
  "id_token": "
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IjFiMTE3cDZ6UUZsQzZUbFNSS1IxcUZmSDBxTSI
sImtpZCI6IjFiMTE3cDZ6UUZsQzZUbFNSS1IxcUZmSDBxTSJ9.eyJpc3MiOiJodHRwczovL2JpbGluZm8ub
mV0IiwiYXVkIjoidGVzdGNsaWVudF9hdXRoY29kZSIsImV4cCI6MTQ4NDU3MjU3MCwibmJmIjoxNDg0NTcy
MjEwLCJpYXQiOjE0ODQ1NzIyMDksInNpZCI6ImZiNDhiOGNmMWZhOGFiZWNmNTM2YjZjZDYxYjcyOGZlIiw
ic3ViIjoiY2xhemFyQGViYXkuY29tIiwiYXV0aF90aW1lIjoxNDg0NTYwODAwLCJpZHAiOiJpZHNydiIsIm
5hbWUiOiJjbGF6YXJAZWJheS5jb20iLCJ1c2VySWQiOiIxMDQyMCIsImFtciI6WyJwYXNzd29yZCJdfQ.AP
8i0i6Yq7V6fDCOdofHv_9RgGV1Xbq4zQ7_gK71v98UJ0LE__4UdSg_y95rbug5nrigiSVJtpb8bsXwLHOJ_
99UEkRvDo6DrymCKc0Mynq7dsW3ZVITm-
P9JiZkGNJBqdG9cEyJH4WkyC1zxoldrEBjUouy6WeFd7979Q2BgYE5P4vWx1-
CRCgrD7AniT63gSkz6ONWHZTrcXBQXo5V8JjH6LOZdqEK1mDF2wxG6A7np1JwpgL1VVpTSdWvkThAMpn7sU
QnODRh56TdiydYgX6zfjmgylHjxSUwjqhyCw2GFGfUnlOLnL6SOhp0V8I2ZrbunQvFUepGvzAW6XkqBA",
  "access_token": "9f92f17660cd0509c2ced6fe7d2d6a8e",
  "expires_in": 360,
  "token_type": "Bearer"
}
```

*Listing 5-3: Authorization Code token retrieval response example*

Note that if you are trying to reuse the same `code` several times, you will get an `invalid_grant` error. You will also receive an error if you try to specify a `redirect_uri` other than the one you specified when obtaining the `code`.

### *Data in the token*

The token holds information about the user, such as name and dealerId. The dealerId is needed for using the CarApi.

## 5.2 Refresh tokens

Bilinfo has support for refresh tokens – tokens that do not expire and can be used to acquire a new access token on behalf of a user.

To get a refresh token, *offline_access* needs to be added as a scope when first requesting a code, an example request is shown in listing 5-5.

```
https://www.bilinfo.net/oauth/connect/authorize
?client_id=yourclientid
&response_type=code
&redirect_uri=http://yourredirecturi
&scope=openid%20profile%20dealer%20https%3A%2F%2Fcarapi.ecgh.dk%2Fauth%2Fwrite%20ht
tps%3A%2F%2Fcarapi.ecgh.dk%2Fauth%2Fread%20offline_access
```

*Listing 5-5: Refresh token example*

The code and client secret must again be exchanged, this time the response contains the refresh token. The request for a refresh token is shown in listing 5-6.

```
POST https://www.bilinfo.net/oauth/connect/token
Content-Type: application/x-www-form-urlencoded

code=codefrompreviousstep&grant_type=authorization_code&client_id=yourclientid&clie
nt_secret=yourclientsecret&redirect_uri=http%3A%2F%2Fautproff.local
```

*Listing 5-6: Refresh token example*

The refresh token should be stored and associated with your current logged-in user. Using the refresh token alongside with your application credentials you will be able to request new identity and access token, as shown in listing 5-7.

```
POST https://www.bilinfo.net/oauth/connect/token
Content-Type: application/x-www-form-urlencoded

refresh_token=refreshtokenspecificfortheuser&grant_type=refresh_token&client_id=you
rclientid&client_secret=yourclientsecret&redirect_uri=yourredirecturi
```

*Listing 5-7: Retrieve access token using refresh token*