



SignaturGruppen



**Nets eID
BROKER**

OpenID Connect Introduction

Version 1.0 2021



Table of Contents

Terminology	3
Changelog	4
Version 1.0	4
Version 0.9	4
Introduction	5
What is OpenID Connect	5
OpenID Connect tokens	5
OIDC Flows	5
Basic OpenID Connect flow	6
Signing authentication requests	6
Signing request parameters	6
Example of Request Object by reference	7
NEB Allow any redirect URI for signed requests	8
Signing Token endpoint requests with private key JWT's	8
Example	8
References	8



Terminology

Term	Description
Nets eID Broker (NEB)	Nets eID Broker. Certified MitID Broker and general broker and identity provider for enterprise services.
OpenID Connect (OIDC)	OpenID Connect 1.0 is an identity layer on top of the OAuth 2.0 protocol
Nets eID Broker Administration web-interface (ADM-UI)	Nets eID Broker Administration web-interface. Interface allowing configuration and administration of the integration



Changelog

Version 1.0

- Updated description of signed requests

Version 0.9

- Created this document. Moved information away from the “Technical Reference” and consolidated the information in this document.



Introduction

This document aims at introducing OpenID Connect with respect to the integration to Nets eID Broker. Simple guides and the most basic functionality will be described in this document.

Refer to the “Nets eID Broker Technical Reference” for a more in-depth technical documentation.

The intended audiences are IT developers and IT architects.

General information, online demonstration, documentation (including newest version of this document) and example code is found at <https://broker.signaturgruppen.dk>.

What is OpenID Connect

OpenID Connect (OIDC) [OIDC] is an identity protocol that utilizes the authorization and authentication mechanisms of OAuth 2.0 [OAuth].

A wide variety of clients may use OIDC to identify end-users, from single-page applications (SPA) to native and mobile apps. It may also be used for Single Sign-On (SSO) across applications. OIDC uses JSON Web Tokens (JWT), HTTP flows and avoids sharing user credentials with services.

OIDC uses OAuth 2.0 as an underlying protocol. The principal extensions are the special scope value “openid”, the use of the extra token, the ID Token, which encapsulates the identity claims in JSON format and the emphasis on authentication rather than authorization.

The OIDC provider performs user authentication, user consent, and token issuance. The client or service requesting a user’s identity is normally called the Relying Party (RP). It can be, for example, a web application, but also a JavaScript application or a mobile app.

Being built on top of OAuth 2.0, OpenID Connect uses tokens to provide a simple identity layer integrated with the underlying authorization framework.

OpenID Connect tokens

The most basic usage of OIDC entails receiving the following tokens.

- **ID Token:** Specific to OIDC, the primary use of this token in JWT format is to provide information about the authentication performed by the end-user. In the context of Nets eID Broker it contains a minimum of information about the end-user and authentication needed to authenticate the end-user. This information, which is provided as claims in the JWT structure of the token, contains information such as subject identifier (sub), issuer, receiver, authentication time, authentication strength.
- **Access Token:** Defined in OAuth2, it provides access to specific user resources as defined in the scope values in the request to the authorization server. The primary usage for access tokens is to access the Userinfo endpoint at the OIDC provider, which provides additional and extended information about the end-user.

All tokens are signed by the OIDC provider.

OIDC Flows

The choice of OpenID Connect flow depends on the type of application and its security requirements. There are three common flows.



- **Implicit Flow:** In this flow, commonly used by SPA's, tokens are returned directly to the RP in a redirect URI. Note that NEB only allows for ID tokens in this scenario. This scenario enables frontend-only applications to authenticate the end-user and avoid any (CORS) API call to the OIDC provider.
- **Authorization Code Flow:** The recommended OIDC flow for most applications. The client application receives an authorization code which then is exchanged to the ID- and access tokens (and optionally additional tokens) via the Token endpoint server to server using a confidential client (entails a client secret).
- **Hybrid Flow:** Combining Implicit and Authorization Code flows, here, the ID Token is returned directly to the RP, but the access token is not. Instead, an authorization code is returned that is exchanged for an access token. As an example, the OWIN framework in .Net requires the receipt of the ID token, and thus does not support the Authorization Code Flow.

Basic OpenID Connect flow

Let's start off with a success and jump right into it!

1: Copy the following URL into a browser:

```
https://pp.netseidbroker.dk/op/connect/authorize?client_id=0a775a87-878c-4b83-abe3-ee29c720c3e7&redirect_uri=https://openidconnect.net/callback&response_type=code&scope=openid mitid
```

2: Complete the login using a MitID or NemID test-user. Information and help for creating a MitID test user is found at the login screen.

3: Follow the guided steps at <https://openidconnect.net>, to exchange the received authorization code to tokens and verify their content.

4: Grab the access token from the output and invoke the Userinfo endpoint manually using Postman or another API client. Information about endpoints and example of request see our Postman collection found via the readme at our .net core demo found here: [Nets eID Broker Demo](#)

Signing authentication requests

Signing request parameters

The OpenID Connect Request Object specifies a **"request"** parameter serialized as a signed and optionally encrypted JWT token holding (most) of the parameters for the flow.

When using the Request Object parameter, most of the parameters will be included inside JWT token instead of passed as separate query parameters in the authorization flow.

Using the Request Object parameter is optional but is recommended for flows with transactional fees, like MitID.

NEB supports both Request Object by value and by reference (i.e., using the **request_uri** parameter).

An option is available to enforce the use of the Request Object parameter for all authentication flows for a client via ADM-UI, enabling a requirement for signed requests for all flows for the client.



Note, that the `client_id` parameters must be set as query parameter when signing requests towards NEB. Also note, that it is perfectly allowed to include any parameter both as a query parameter as well as included in the signed JWT. The values specified in the signed JWT will be used.

Signing the Request Object parameter must be done with one of the configured client secrets.

See [ROBJ] for reference and specification.

Example of Request Object by reference

An example of a valid authentication request with a Request Object parameter:

```
client_id: bcc7595b-2ed0-445d-a507-42ca21382877
request:
eyJhbGciOiJIUzI1NiIsImtpZCI6bnVsbCwidHlwIjoiaSldUIn0.eyJjbGllbnRfaWQiOiJiY2M3NTk1Yi0yZWQwLTQ0NWQtYTUwNy00MmNhMjEzODI0NzciLCJyZWVpcVjdf91cmkiOiJodHRwczovL2xvY2FsaG9zdDolMDE1L3NpZ25pbilvaWRjIiwicmVzcg9uc2VfdHlwZSI6ImNvZGUlLCJjb2RlX2NoYWxsZW5nZSI6ImlaWFppX3RMbWhrc2FSZVdsR0JqbERSU1FuLXdrWUVsdXVsak0zbDZRSkUiLCJjb2RlX2NoYWxsZW5nZV9tZXRob2QiOiJTMjU2IiwicmVzcg9uc2VfbW9kZSI6ImZvcmlfcG9zdCIsIm5vbmNlIjoiaNjM3NTcwMjAwNDg3MjI4MTg2LlppqUmpOV1V4TWpjde1tWXdz eTAwTjJZeExRXlZelF0T0RBek5qY3hZbVV4T0dFek16UTJOak13Wm1FdE1HUXl0QzAwTVdSaE xXRTJZVEV0TmlVelpEYzNaV1E0WVRabCIsInNjb3BlIjoib3BlbmlkIiwiaWRwX3ZhbHVlcyl6 IiIsImkcF9wYXJhbXMiOiJ7XCJtaXRpZfwiOntcImVuYWJsZV9hcHBfc3dpdGNoXCI6ZmFsc2 UsXCJlbnFibGVfc3RlcF91cFwiOmZhbHNlfSxcIm1pdGlkX2RlbW9cIjpp7XCJlbnFibGVfYXBw X3N3aXRjaFwiOmZhbHNlLFwiZW5hYmxlX3N0ZXBfdXBcIjpmYWxzZX0sXCJuzW1pZfwiOntcIn ByaXZhdGVfdG9fYnVzaW5lc3NcIjpmYWxzZX19IiwiaXVkiJoiaHR0cHM6Ly9icm9rZXJkZXlYU c2lnbmF0dXJncnVvcGVuLmRrL29wIiwiaXhwIjoiaMTYyMTQyMzU0OCIsImlzcyl6ImJjYzclOT ViLTJlZDAtNDQlZC1hNTA3LTQyY2EyMTM4Mjg3NyIsImhhdCI6IjE2MjE0MjMyNDgiLCJmYmYi OiIxnNjIxnNDIzMjQ4In0.dHgZoyUTgiFBr_Y57whuzPGknorl5C-Jh_kZboN_OlA
```

Note that the example contains optional parameters, like the PKCE and identity provider specific parameters included here.

```
{
  "client_id": "bcc7595b-2ed0-445d-a507-42ca21382877",
  "redirect_uri": "https://localhost:5015/signin-oidc",
  "response_type": "code",
  "code_challenge": "izXZi_tLmhksaReWlGBjlDRSQn-wkYEluuljM316QJE",
  "code_challenge_method": "S256",
  "response_mode": "form_post",
  "nonce":
  "637570200487228186.ZjRjNWUxMjctMmYwYy00N2YxLWEyYzQtODAzNjcxYmUxOGEzMzQ2Nj
  MwZmEtMGQyNC00MWRhLWE2YEtNmUzZDc3ZWQ4YTZl",
  "scope": "openid",
  "idp_values": "",
  "idp_params":
  "{\"mitid\":{\"enable_app_switch\":false,\"enable_step_up\":false},\"mitid
  demo\":{\"enable_app_switch\":false,\"enable_step_up\":false},\"nemid\":{\"
  private_to_business\":false}}",
  "aud": "https://brokerdev.signaturgruppen.dk/op",
  "exp": "1621423548",
  "iss": "bcc7595b-2ed0-445d-a507-42ca21382877",
  "iat": "1621423248",
  "nbf": "1621423248"
}
```



NEB Allow any redirect URI for signed requests

For signed requests, the client can specify any redirect URI (the `redirect_uri` authentication parameter). This enables a more dynamic use of the protocol and avoids the whitelist approach normally used.

Signing Token endpoint requests with private key JWT's

The OpenID Connect specification recommends a client authentication method based on asymmetric keys. With this approach, instead of transmitting the shared secret over the network, the client creates a JWT and signs it with its private key.

More information is found in the Client Authentication section of the OIDC specification found in [CLIENT-AUTHENTICATION].

Example

A private key JWT is prepared and signed with the following format

```
{
  "jti": "ec454782-93c1-4160-8e46-502532df52f2",
  "sub": "bcc7595b-2ed0-445d-a507-42ca21382877",
  "iat": 1619296461,
  "nbf": 1619296461,
  "exp": 1619296581,
  "iss": "bcc7595b-2ed0-445d-a507-42ca21382877",
  "aud": "[authorityUrl]/connect/token"
}
```

And heres a curl example of the post request:

```
curl --location --request POST 'https://localhost:5001/connect/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=authorization_code' \
--data-urlencode 'client_id=asd...ca21382877' \
--data-urlencode 'redirect_uri=<redirectUri>' \
--data-urlencode 'code=35CA4BC2A0F5DB9A...1BEF00150EEA44' \
--data-urlencode 'client_assertion=...XahMP78S7DdgwZEvVa4vA3usNsJg' \
--data-urlencode 'client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer'
```

More details for .Net can be found here:

<https://docs.duendesoftware.com/identityserver/v5/tokens/authentication/jwt/>.

References

1. [OIDC WEB] "OpenID Connect Home": <https://openid.net/connect/>
2. [OIDC] "OpenID Connect core": https://openid.net/specs/openid-connect-core-1_0.html
3. [OAuth] "The OAuth 2.0 Authorization Framework": <https://tools.ietf.org/html/rfc6749>