This document is under construction and subject to change. Some of the functionality described in this document, is not available yet.

Nets eID BROKER

Technical reference

for service providers

Version 0.9.6.4 2020

SignaturGruppen

Table of Contents

Terminology

Term	Description
Nets eID Broker (**NEB**)	Nets eID Broker. Certified MitID Broker and general broker and identity provider for enterprise services.
Nets eID Broker Administration web-interface (**ADM-UI**)	Nets eID Broker Administration web-interface. Interface allowing configuration and administration of the integration
OpenID Connect (**OIDC**)	**OpenID Connect** 1.0 is an identity layer on top of the OAuth 2.0 protocol
OAuth	OAuth 2.0 is the industry-standard protocol for authorization. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for web applications, desktop applications, mobile phones, and living room devices. This specification and its extensions are being developed within the IETF OAuth Working Group.
JWT (JSON Web Token)	JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties.
JWS	JSON Web Signature (JWS) represents content secured with digital signatures or Message Authentication Codes (MACs) using JSON-based data structures. Cryptographic algorithms and identifiers for use with this specification are described in the separate JSON Web Algorithms (JWA) specification and an IANA registry defined by that specification. Related encryption capabilities are described in the separate JSON Web Encryption (JWE) specification.
JWE	JSON Web Encryption (JWE) represents encrypted content using JSON-based data structures. Cryptographic algorithms and identifiers for use with this specification are described in the separate JSON Web Algorithms (JWA) specification and IANA registries defined by that specification. Related digital signature and Message Authentication Code (MAC) capabilities are described in the separate JSON Web Signature (JWS) specification.
National Standard for Identiteters Sikringsniveauer (**NSIS**)	Collaboration for trust to digital identities and digital identity-services in Denmark.
MitID	National identity and authentication solution in Denmark.
NemID	National identity and authentication solution in Denmark. Is being replaced by MitID in 2021.
MitID Broker	Certified MitID Identity Broker. Trusted part of the MitID ecosystem.
NemLog-In3 (**NL3**)	NL3 plays a central role in Denmark's digital infrastructure by making it possible for Danish

	citizens and companies to log in to public self service solutions. NL3 provides the CA services for the Oces3 PKI.
Offentlige Certifikater til Elektronisk Service (**OCES**)	OCES-standarden / The OCES-standard. OCES is a Danish public standard for “Public Certificates for Electronic Service”.
VOCES3	Company certificate issued under the OCES3 standard by the NemLog-In3 CA.

Introduction

This document describes the technical integration with the Nets eID Broker (NEB) and should be considered the primary resource when service providers integrate with the NEB.

The intended audiences are IT developers and IT architects.

Business functionality specified in this document may be subject to different commercial agreement requirements.

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

Integrating to the Nets eID Broker overview

This section is meant as a way for service providers to gain a quick overview of the technical requirements and development effort required to integrate with the NEB.

OpenID Connect

OpenID Connect (OIDC) is the primary protocol used when integrating with NEB. It allows almost all types of clients to integrate to NEB and supports for complex client scenarios like mobile apps.

OpenID Connect is fully supported and thus enables the widest range of clients to benefit from the services offered including, but not limited to, legacy OAuth clients, mobile apps, CRM portals like Microsoft Dynamics and Single Page Applications (SPA) written in JavaScript (client application).

It allows for any programming language to integrate via official OIDC patterns, by developing the integration or by using the examples and demos given as part of the documentation for the NEB.

OpenID Connect supports a strong security model, while retaining a large flexibility to support various flows and clients.

Enterprise features like Single Sign On/Out, session management, API authorization, Long Lived Sessions (using refresh tokens) and automatic discovery of services, endpoints and cryptographic material is all accessible through the NEB platform.

Administration web-interface and API

[Note: Not available for service providers in current version]

All administration and configuration for service providers is handled through the **Administration web-interface (ADM-UI)**, which allows service providers to configure their own services and clients, setup test-users, configure cryptographic settings like generating or uploading secrets for their clients and API resources, view and search logs etc.

ADM-UI will be the entry point for all configuration and setup of the integration and will provide a way to see logs, statistics, and other relevant information.

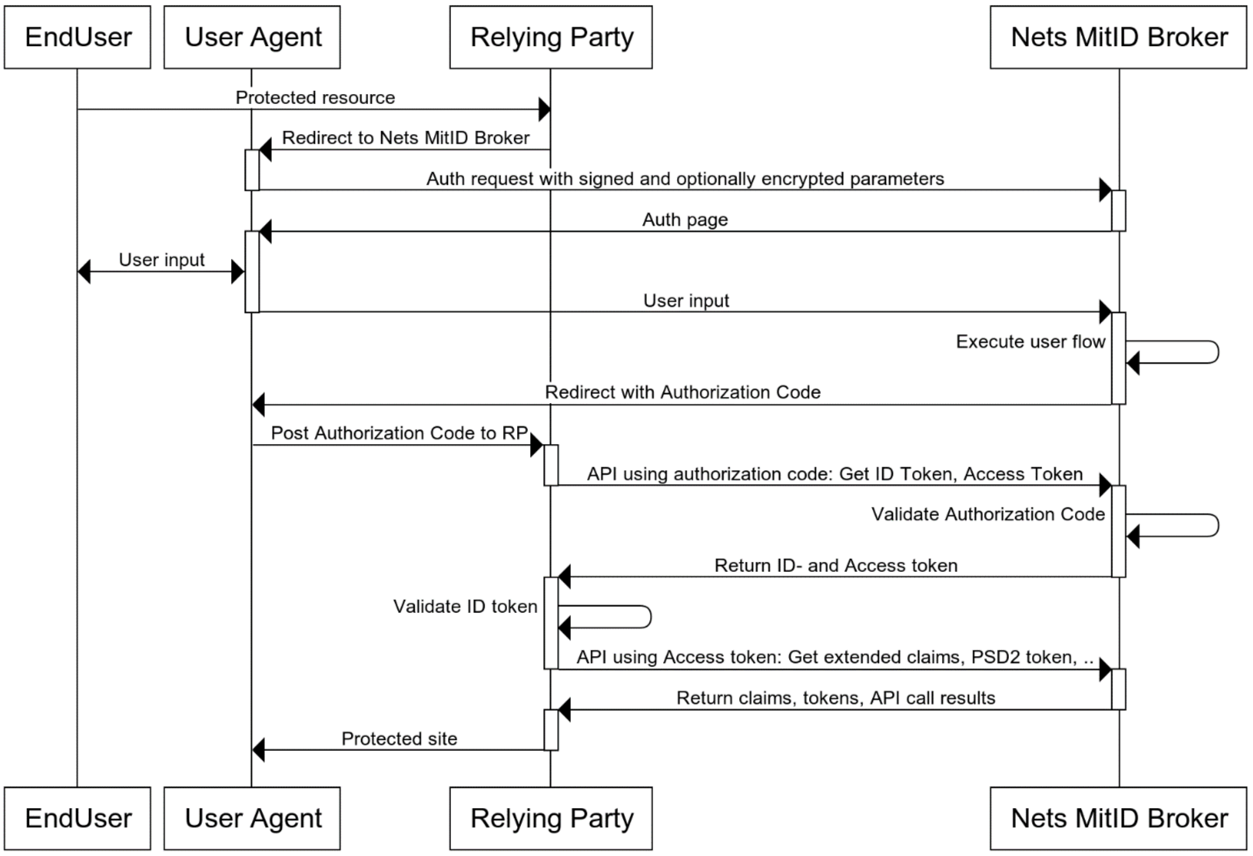
Web and app client integration

This section describes the overall integration for client applications for the NEB.

OpenID Connect

The NEB is integrated with the service providers web pages and apps using OpenID Connect.

An example of a user-flow is illustrated here.


EndUser
User Agent
Relying Party
Nets MitID Broker
Protected resource
Redirect to Nets MitID Broker
Auth request with signed and optionally encrypted parameters
Auth page
User input
User input
Execute user flow
Redirect with Authorization Code
Post Authorization Code to RP
API using authorization code: Get ID Token, Access Token
Validate Authorization Code
Return ID- and Access token
Validate ID token
API using Access token: Get extended claims, PSD2 token, ..
Return claims, tokens, API call results
Protected site

The flow is conceptually the same in both full-page redirect and pop-up variants and is started by redirecting the user to the NEB authorization endpoint with the required request parameters.

For the Authorization Code flow, a secure and preregistered client is required for allowing the exchange of the authorization code for ID- and access tokens. For the rest of the API calls, the access token (which has an expiry) is used for authorization from the service provider to the NEB.

The OIDC client used by the service provider has one or more preregistered 'login URLs', which determines where the user will be returned to with the authorization code. The same OIDC client will typically have one or more client secrets, protecting all API calls from the client to the NEB, including the exchange of the authorization code to tokens.

OpenID Connect client

An OpenID Connect flow is initiated by a secure and preconfigured client. A client in this context, is a unique configuration specifying the allowed flows, available endpoints, APIs, features etc.

When a service integrates with the NEB, one or more clients defines the technical integration from the service providers services to the NEB platform.

The clients are created and configured in ADM-UI and is used directly by the integrating systems when interfacing with the OpenID Connect specification.

A client consists of (non-exhaustive list):

- Client ID: unique identifier

- Client Secrets: Symmetric or asymmetric keys for communication
- Redirect URL list: List of approved URLs for flow control
- Logout URL list: List of approved URLs for flow control

Behind the scenes, a client is mapped to an allowed set of features, scopes, and parameters that the specific client can include in an authorization request. The flow will fail if a client requests anything not whitelisted for the client.

Specific flow control configuration like encryption requirement for requests is configured through ADM-UI and is tied to the client but will not be part of the integrating client values. Note that most settings tied to the client exists only through ADM-UI and can be updated dynamically without changing the integrating systems configuration of the client.

Access to APIs is done through the same mechanisms by using the access tokens from supported user flows or by directly getting access to APIs by getting specific API access tokens from the Token Endpoint (secured by client secrets and configured access).

ADM-UI handles all this configuration and allows the generation of a JSON file with a valid OpenID Connect configuration for a client making it easy to hand out the configured client configuration to projects or for direct use with one of the supplied integrations clients.

OpenID Connect Authorization Code flow

This describes the basics of an OpenID Connect flow with NemID or MitID using NEB.

1. The end user accesses the service provider site with a request to log on.
2. The end user browser is redirected to NEB to begin identification. Sample identification request: **https://netsbroker.mitid.dk/op/connect/authorize?client_id=<client_id>&response_type=code &redirect_uri=<redirect_uri>&scope=openid mitid ssn&state=<state>&nonce=<nonce>acr_values=**https://data.gov.dk/concept/core/nsis/Low**&idp_values=mitid**
3. End user identification is initiated towards a selected eID. The end user supplies his/her credentials.
4. NEB redirects the end user to the service provider **redirect_uri** by appending the query string ?code=<authorization code>. Similarly, the nonce and state parameters, as sent by the customer, are also appended to the **redirect_uri**.
5. The service provider (backend) requests the ID-, access- (and additional optional) tokens based on the authorization code.

Example requests

(URL encoding removed, and line breaks added for readability)

Authorization request

```
GET /connect/authorize?
    client_id=client1&
    scope=openid mitid&
    response_type=code&
    redirect_uri=https://myapp/callback&
    state=abc&
    nonce=xyz
```

Authorization Code Token endpoint example

```
POST /connect/token
```

```
client_id=client1&
client_secret=secret&
grant_type=authorization_code&
code=hdh922&
redirect_uri=https://myapp.com/callback
```

Secure vs. public OpenID Connect client

If an OpenID Connect client is configured with a client secret and is required to use this secret when communicating with the Token endpoint, the client is considered a secure client. If the client has no client secret and can retrieve tokens from the Token endpoint without a client secret, it is considered a public client.

Not all clients are configured as a secure client as some applications, like mobile apps, are inherently public and it does not make sense to share a secret across all instances of the mobile application. Instead the security is based on control on the redirect URL and by using the OIDC PKCE extension.

If able, an application should always have a secure client controlled by a backend application – but in some scenarios the OIDC integration is done client-side.

The NEB platform supports all variants and the security for all models can be tailored to fit the application in question.

Code, Hybrid and Implicit flows

The recommended OIDC flow is the Authorization Code flow, which does only communicate the issued tokens via the Token endpoint (backend to backend).

If the ID token or the access token is required in the user redirect response, the Hybrid or Implicit flows is supported.

This is configured in ADM-UI and supports allowing a client to request the ID token or access token in the redirect response.

See [OIDC] for reference.

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.

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

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

Encrypting the Request Object parameter can be done with a specific key configured for the client.

See [ROBJ] for reference and specification.

User flows

User flows represent the UI flow and experience the end-user will see and interact with when authenticating, giving consent etc.

This section describes the general integration and setup for end-user flows for services provided by the NEB platform.

User flow parameters

This section describes how to control and select various parameters for setting up and controlling the user flow.

This section will cover the structure used by the rest of this document for the request and result for user flows as well as the general parameters applicable for all user flows.

Required authorization request parameters:

Parameter	Description
client_id	Identifier of the client.
response_type	Authorization Code flow, Hybrid flow and Implicit flows are supported, as specified in [OIDC].
redirect_uri	URI that the response will be sent to when authentication is finished. Must be from a list of preregistered URLs for this client.
scope	Space separated list of scopes that client is requesting authorization for. **openid** must be included for all requests.

Supported authorization request parameters are listed below. See [OIDC] for reference.

Parameter	Description
nonce	String value used to associate a client session with an ID Token, and to mitigate replay attacks. Must be unique per request per client.
state	Opaque value used to maintain state between the request and the callback.
request	This parameter enables OpenID Connect requests to be passed in a single, self-contained parameter and to be optionally signed and/or encrypted. It represents the request as a JWT whose claims are the request parameters. It is recommended for clients able to start flows with associated fees. This parameter can be made mandatory for a client in ADM-UI. See section 6 in [OIDC].
request_uri	This parameter enables OpenID Connect requests to be passed by reference, rather than by value. The **request_uri** value is a URL using the https scheme

	referencing a resource containing a Request Object value, which is a JWT containing the request parameters. This parameter can be made mandatory for a client in ADM-UI. See section 6 in [OIDC].
language	Sets the end-user language. Possible values are • **da** (Danish) • **en** (English) • **kl** (Greenlandic)
max_age	Maximum Authentication Age. Specifies the allowable elapsed time in seconds since the last time the end-user was actively authenticated by NEB. If the elapsed time is greater than this value, NEB will attempt to actively re-authenticate the end-user
prompt	Space delimited, case sensitive list of ASCII string values that specifies whether the Authorization Server prompts the end-user for reauthentication. Possible values are • **none** (default) • **login** (force full login) • **select_account** (allows change of identity) If the **select_account** value is specified. The end-user will be allowed to change the active selected identity for the requested service. If the end-user already has signed in with the desired identity in the SSO context of the requesting service, the end-user will be able to select the identity without the need of re-entering credentials. If **login** is used, the end-user will be forced to complete the requested authentication flow. Refer to the [OIDC] for reference.

Supported identity provider parameters. See the Identity Providers section for available options.

Parameter	Description
idp_params	Identity provider parameters. Custom parameter supported by NEB to enable customization of the specific identity provider flows. See the Identity Providers section for reference. The **idp_params** parameter value is represented in an OAuth 2.0 request as UTF-8 encoded JSON (which ends up being form-url-encoded when passed as an OAuth parameter). When used in a

	Request Object value, the JSON is used as the value of the **idp_params** member.
acr_values	Requested Authentication Context Class Reference values. Space-separated string that specifies the **acr** values that the NEB is being requested to use for processing this authentication request, with the values appearing in order of preference.
ial_values	Identity Assurance Level values. Space-separated string that specifies the **ial** values that the NEB is being requested to use for processing this authentication request, with the values appearing in order of preference.
amr_values	Requested Authentication Method References values. Space-separated string that specifies the **amr** values that the NEB is being requested to use for processing this authentication request, with the values appearing in order of preference. If not set, no restrictions for the choice of authentication type will be enforced for the end-user.
idp_values	Identity provider list. Space-separated string that specifies the **idp** values that the NEB is being requested to use for processing this authentication request, with the values appearing in order of preference. If not set, all possible identity providers configured for the client will be made available to the end-user.
identitytype_values	Identity type. Space-separated string that specifies the **identitytype** values that the NEB is being requested to use for processing this authentication request, with the values appearing in order of preference. If not set, no restrictions for the selected identity type will be enforced for the end-user.

Reauthentication

As a general mechanism, NEB will always try to optimize the user experience and steps required from the end-user for an authentication flow. If a user has an active session with NEB and enters a new authentication flow, the existing session will under certain conditions be re-usable and only trigger additional actions needed from the end-user to fulfill the requested parameters.

Step-up

If an end-user has an active session at NEB and a new authentication flow is initiated requesting a higher **acr** value, the end-user will *may* enter a step-up flow. This is automatically handled by NEB based on the active session and the requested authentication.

Note, that step-up flows only make sense when requesting a reauthentication from the same identity provider.

An example of a step-up flow would be sending the user for authentication to a higher NSIS Level of Assurance, e.g. the user earlier authenticated for NSIS Low and then a new authorization request is made for NSIS Substantial.

The user will automatically enter a step-up flow (unless requested otherwise) letting the user select the appropriate electronic identification device to complete the step-up.

This way the service only needs to express what the result should be without having to take previous flows and authentications into consideration.

Requesting additional scopes

When requesting additional scopes for an existing user session only the required verifications and user-steps are invoked.

If the requesting client can request the additional scope, the existing session will be reused to issue a new session with the requested scopes. If the scope triggers end-user interaction, like a consent-flow, the user will be prompted for action, but the user will not have to reauthenticate unless required.

Authentication Error Response

If the end-user denies the request or the end-user authentication fails, NEB informs the service provider (client) by using the error response parameters defined in Section 4.1.2.1 of [OAuth].

The error response will honor the **response_mode** authorization request parameter and thus support to be sent back to the service as a GET (default) or POST request.

Note, that the end-user will only be redirect back to the service provider (client) if the authorization request is valid. If the authorization request is invalid, the end-user will be presented a generic error response at NEB and will not be redirected back to the service provider (client).

A list of error codes and descriptions will be made available in this document.

Scopes

Scopes are passed as a space separated list of string values in the **scope** authorization request parameter and determine the requested authorizations as well as the requested user claims.

A scope has the following functions

- Maps to a list of user claims for the ID token and UserInfo endpoint
- Is mapped directly to the issued access token **scope** claim.
- Grants authorization for API access by setting relevant values in the access token **aud** claim.
- Some scopes trigger certain user flows or end-user actions such as required end-user consents.
- Client must be allowed to use the scopes requested.

Environments

This section defines the available environments available from NEB and their respective URLs and certificates.

Security related information like TLS and VOCES3 signing certificate DN and CA root information, will also be available for each environment.

Pilot test environment

Variable	Values

Authority URL	https://brokertest.signaturgruppen.dk/op
Discovery endpoint	https://brokertest.signaturgruppen.dk/op/.well-known/openid-configuration

Nets eID Broker APIs

The OpenAPI/Swagger documentation is not released yet.

NEB exposes several APIs. This document describes the principles behind how the APIs work.

All APIs are provided as REST APIs, exposed over HTTPS (HTTP/1.1 protected by TLS 1.2 or higher).

The exposed APIs are:

- **Broker API**: Identity based APIs supporting authentication and authorizations, including OpenID Connect endpoints and the CPR-Match API. The Broker API is published partially through the Swagger specification and partially through the OpenID Connect Discovery endpoint.
- **Administration API**: The administration API supporting all administrative and support functionalities also available via ADM-UI. Use of this API is optional.
- **Privilege API**: Supporting a privilege API available for all Service Providers as a stand-alone service. Used internally by NEB for all relevant services. This enables service to setup roles and permissions across internal and external services. Compatible with and based on the OIO Basic Privilege Profile [OIO PRIV]

API specification

All APIs are specified according to the OpenAPI 3.0 specification (previously known as “Swagger”). In practice this means that the APIs are described in machine-readable YAML documents, describing the resources exposed in the APIs, the available methods etc. as well as human-readable descriptions of the API. The YAML files can be then be used by tooling to create API-specific clients and stubs or be used as input into tools for API testing. They can also be used for generating documentation. The API documentation is delivered in the form of HTML files generated from the YAML specifications

Error Handling

Errors are reported using HTTP error codes. Each API function documents the error codes that may be returned. In addition, a JSON error object is returned that provides further information on the error. The structure of this error object is described in the YAML files for the specific API.

API Versioning and Backwards Compatibility

Whenever an API is updated, a new version of the specification and the documentation is published. All API specifications are versioned. The guiding principle for the evolution of the API is that all updates are made to avoid “breaking changes”, i.e. changes that cause problems for clients that were developed according to previous versions. Thus, new functionality is mainly exposed as new resources or new attributes/fields in existing data structures. Only if imperative, to ensure the security or future maintainability, breaking API changes will be introduced. Because of this principle, users of the API are required to ignore fields in data structures that they don’t recognize, unless otherwise noted in the documentation. Furthermore, the users of the API are to rely only on documented behavior, and to ignore absence of resources or functionality that is not documented. The documentation will state documented behavior as requirements.

In the documentation APIs are versioned as a semantic versioning scheme (“major.minor.revision”). Breaking changes are signaled by increasing the major version number. This is expected to be a rare occurrence after the development phase is over. If only the minor or the revision number is updated, existing clients targeting the major version number will be compatible with the new version of the API. The URL’s exposed by the API contain, as their first sub-resource, a version number which reflects the major version. This is initially “v1”, reflecting the first version of the API. If no breaking changes are introduced, this number will stay at “v1.

OAuth 2.0 Authorization Framework

All APIs are protected using the OAuth 2.0 authorization framework [OAuth].

The Token endpoint is the entry point for getting access- or service tokens issued, which is then used as authorization bearer tokens.

Access tokens are retrieved from end-user authentication flows or via the “Client Credentials Grant” type flows at the Token endpoint. Service tokens are always retrieved from the Token endpoint using the “Client Credentials Grant” type flow.

Broker API – OpenID Connect endpoints

In this section the available OpenID Connect endpoints will be listed.

All listed endpoints in this section will conform to the OpenID Connect specification and will be listed in the Discovery endpoint.

Discovery endpoint

The “OpenID Connect Discovery” endpoint. See [OIDC-DISC] for reference.

NEB uses OpenID Connect Discovery which allows for automatic retrieval and dynamic changes of endpoints, cryptographic primitives, supported scopes and other features.

UserInfo endpoint

For most user authentication flows, the resulting access token provides access to the UserInfo endpoint. The endpoint is specified in the Discovery endpoint and returns the full user-claims list issued for the end user for the authentication context.

For identity scopes, user-claims will be available via the UserInfo endpoint. Some specific user-claims is also available in the ID token – these are explicitly described in the relevant identity provider section in this document.

A result from the User Info endpoint after a successful MitID authentication flow with scope=”openid mitid ssn” could look like this:

```
{
 “sub”                                 : "bab646bb-8608-4ac7-ac42-cee4ad490600",
 “mitid_uuid”                          : “af0196a3-6c61-464d-ab04-6394191a753d”,
 “mitid.age”                           : “32”,
 “mitid.ial_identity_assurance_level”  : “MEDIUM”,
 “da.cpr”                              : “12345678-1111”,
 “mitid.cpr_name”                      : “Hans Hansen”,
}
```

The response format of the UserInfo endpoint can be configured in ADM-UI to enable a signed and/or encrypted response as specified in [OIDC] section 5.3.2.

Issued tokens

This section describes the available tokens issued by NEB. All tokens are issued via the **Token endpoint** by the **Code Authorization Grant** or the **Client Credentials Grant**.

Some integrations will get tokens via the end-user browser (Hybrid- and Implicit flows).

User flow authentication result

A user flow results in one or more of the following tokens

- **ID token**: Describing the authenticated end-user.
- **Access token**: Providing access to configured endpoints on behalf of the end-user
- **Service token (optional)**: Token for one or more specific APIs.
- **Refresh token (optional)**: Providing ability to maintain a long-lived session by re-acquiring access tokens using the refresh token.
- **Userinfo token (optional)**: Provides all user claims requested in a single JWT.
- **Transaction token (optional)**: Provides a self-contained sealed record of the transaction completed by the end-user.

ID-, access-, service and transaction tokens comply with the [JWT] specification.

Access- and service tokens are not meaningful outside the audience of the token. Access tokens can be configured to include access and authorization for both internal and external REST APIs. Often the access token is used for accessing the UserInfo endpoint at NEB.

Refresh tokens are opaque and thus not meaningful outside the scope of NEB. See the OpenID Connect specification for reference on how to use the refresh tokens.

The transaction token will be available as a sealed (signed by a Nets eID Broker OCES3 organization certificate) record of the end-user completed transaction.

The transaction token response is self-contained and cryptographically sealed record suitable for long-term storage and as a proof-of-transaction.

The transaction token includes the relevant authentication information, a unique transaction identifier as well as relevant transaction specific information like end-user approved text linked to the transaction.

Note that the transaction token is optional and should only be requested if required for internal revision or similar requirements.

NEB does not currently support encrypted tokens.

Client Credentials (backend to backend)

Services can get tokens from the Token endpoint by authenticating directly using their client credentials and allows for issuing tokens for services directly.

The following tokens can be issued:

- **Service token**: Token for one or more specific APIs.

ID token

Note, that the ID token does not include all issued user claims. The full list of user claims will be available from the UserInfo endpoint or in the Userinfo token.

ID tokens issued include the claims listed below with values as specified.

Claim	**Value**
iss	Identifier for the issuer as an URL using https scheme.
jti	A unique identifier for the token, which can be used to prevent reuse of the token.
sid	Session ID. String identifier for a Session. This represents a session of a user agent or device for a logged-in end-user at a service provider. Different **sid** values are used to identify distinct sessions at NEB. The **sid** value need only be unique in the context of a particular issuer. Its contents are opaque to the service provider. Its syntax is the same as an OAuth 2.0 Client Identifier.
sub	NEB specific UUID representing the authenticated end-user. Primary end-user identifier.
aud	Audience(s) that this ID Token is intended for.
exp	Expiration time on or after which the ID Token MUST NOT be accepted for processing.
iat	Time at which the JWT was issued. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.
auth_time	Time when the end-user authentication occurred. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.
nonce	String value used to associate a Client session with an ID Token, and to mitigate replay attacks. The value is passed through unmodified from the Authentication Request to the ID Token.
amr	Authentication method reference
acr	Authentication Context Class Reference. Represents the Authenticated Level of Assurance (LoA). The [NSIS] framework forms the basis for how NEB handles LoA for authentications, but the available **acr** values are not restricted to the values specified in [NSIS]. Refer to the specific identity providers for a list of possible **acr** values.
ial	Identity Assurance Level Strength of the Identity registration process.

	Identity Assurance Level can be higher than the acr/LoA value, as the identity can be enrolled/registered at a higher level, than the achieved authentication level for a specific session. If the ial is not known, this value is not set.
idp	Identity provider who issued the underlying identity. Refer to the specific identity providers for a list of possible **idp** values.
identitytype	One of • private • professional • test
transactionid	Transaction ID. Unique identifier for the completed (end-user) transaction. Distinct identifiers are used to identify different transactions completed at NEB.
spec_ver	0.9 for this version. (Still under development)

Example of an ID token payload (decoded) from a MitID Identity Provider:

```
{
 "sub"                                  : "bab646bb-8608-4ac7-ac42-cee4ad490600",
 "mitid.uuid"                           : "7027a386-aa7c-4dd6-93de-ebffd670f8b5",
 "mitid.ial_identity_assurance_level"   : "MEDIUM",
 "iss"                                  : "https://netsbroker.mitid.dk",
 "jti"                                  : "5964f27b-7a7c-4f3d-99fe-aee934f03397",
 "aud"                                  : "9ad129c2-0341-40e4-a184-b834272217dd",
 "nonce"                                : "3f0fc970-9727-4b3f-9f30-78793487ac7b",
 "auth_time"                            : 1311261123,
 "acr"                                  : "https://data.gov.dk/concept/core/nsis/Low",
 "amr"                                  : "mitid.password",
 "identitytype"                         : "private",
 "idp"                                  : “mitid”,
 "iat"                                  : 1311290550,
 "exp"                                  : 1311291550,
 "spec_ver"                             : “0.9”
}
```

Default expiry for ID tokens is 5 minutes.

Access token

The resulting access tokens authorizes the bearer on the behalf of the user. Unless otherwise configured for the client, the resulting access token provides authorization for the NEB UserInfo endpoint resulting in the full list of claims issued to the user for the authentication in question.

Depending on configuration, capabilities, roles, permissions and granted access for the client, the access token can authorize the client on behalf of the user to

- Access specified APIs from NEB, like the Privilege API.
- Access internal APIs (i.e. internal to the Organization/Service in question)
- Access external APIs (exchange **access token** for a **service token** at the Token endpoint)

Default expiry for access tokens is 1 hour.

Service token

If allowed, a service can exchange the retrieved access token for a service token at the Token endpoint or retrieve a service token using the Client Credentials Grant.

Service tokens is simply bearer authorization tokens for specific APIs protected by the NEB infrastructure.

Configuration and permissions for service tokens are setup via the ADM-UI and allows authorization to external APIs on behalf of the end-user or service in question.

ADM-UI enables configuration of permissions between organizations and services in this way using the Privilege API backing the NEB infrastructure.

Default expiry for service tokens is 1 hour.

Refresh token

If the client is allowed for long-lived sessions, refresh tokens are issued when requesting the **offline_access** scope.

Refresh token expiry and other properties are configured in ADM-UI.

Userinfo token

The userinfo token includes all user claims in a single signed JWT including, including the full Userinfo endpoint response list of claims.

This allows the retrieval of all user claims directly from the Token endpoint in a signed format.

The Userinfo token is requested by setting the scope value **userinfo_token**.

Transaction token

The transaction token will contain additional claims based on the end-user identity provider and provided identity provider parameters. These additional claims are explained under the specific identity provider in this document.

It is not designed to replace the ID token and will most often contain sensitive information.

Transaction tokens are meant as a receipt for the completed end-user transaction. The token is signed by a special signing key and formatted to support long-term verification of the end-user transaction.

Many of the issued claims, are the same as found in the accompanied ID token.

In this section all the claims that are always present in transaction tokens are specified. In addition, each identity provider will have its own set of claims that can be included depending the context.

Transaction tokens will be set in the Token endpoint response, if configured for the client and if requested using the **transaction_token** scope.

An accompanying OCSP revocation check response for the signing certificate, will be set in the Token endpoint response, formatted as a UTF-8+Base64 encoded string. The **signing_cert_ocsp_nonce** claim set in the transaction token is the nonce used for the OCSP response.

Token endpoint response:

```
{
  “id_token”: “AGGSSDDhY3Rpb…AFGGRh”,
  …,
  "transaction_token": “VHJhbnNhY3Rpb…BEYXRh”,
  “transaction_token_ocsp_resp”: “VHJhbnNhY3Rpb2…UZXN0IERhdGE=”
}
```

Transaction tokens issued always include the following claims:

Claim	**Value**
iss	Identifier for the issuer as an URL using https scheme.
sub	NEB specific UUID representing the authenticated end-user. Primary end-user identifier.
iat	Time at which the JWT was issued. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.
auth_time	Time when the end-user authentication occurred. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time.
nonce	String value used to associate a client session with a transaction token, and to mitigate replay attacks. The value is passed through unmodified from the authentication request to the transaction token. This is the same nonce as set in the ID token.
signing_cert_ocsp_nonce	Nonce for signing certificate OCSP revocation status check. UTF-8+Base64 encoded string. Nonce used for checking OCSP revocation status for the transaction token signing certificate after the token has been signed.
amr	Authentication method reference
acr	Authentication Context Class Reference. Represents the Authenticated Level of Assurance (LoA). The [NSIS] framework forms the basis for how NEB handles LoA for authentications, but the available **acr** values are not restricted to the values specified in [NSIS]. Refer to the specific identity providers for a list of possible **acr** values.
ial	Identity Assurance Level Strength of the Identity registration process. Identity Assurance Level can be higher than the acr/LoA value, as the identity can be

	enrolled/registered at a higher level, than the achieved authentication level for a specific session. If the ial is not known, this value is not set.
idp	Identity provider who issued the underlying identity. Refer to the specific identity providers for a list of possible **idp** values.
identitytype	One of • private • professional • test
transactionid	Transaction ID. Unique identifier for the completed (end-user) transaction. Distinct identifiers are used to identity different transactions completed at NEB.
recipient_info	Type: JSON. The top-level members of the **recipient_info** JSON object are: • **organization.number** • **organization.name** • **organization.country** • **redirect_uri**: The URL that the end-user is redirect to from NEB upon successful completing the transaction. The **recipient_info** parameter value is represented as UTF-8 encoded JSON.
spec_ver	0.9 for this version. (Still under development)

The receiving party should store both the transaction token and the OCSP response. Optionally, the signing certificate can be stored.

Security

This section will cover supported cryptographic algorithms, supported TLS versions, certificate pinning and other security related issues.

OpenID Connect provides a high level of security, but for some application require additional security hardening. This section will cover the available options.

All algorithms specified in this section is specified in [JWA].

PKCE

Proof Key for Code Exchange by OAuth Public Clients (PKCE) is an extension to the Authorization Code flow to prevent certain attacks and to be able to securely perform the OAuth exchange from public clients.

This is prevalent and recommended when integrating to NEB from a mobile (Android and iOS) platform and allows the initiating app to be the only one who is able to retrieve the issued tokens, even though the client is a public client.

PKCE is fully supported. See [PKCE] for reference.

TLS certificate authorities

This section will include the list of publicly trusted CAs issuing certificates for all NEB endpoints.

This information can be used to pin trust for the CAs issuing the TLS certificates used by the NEB platform.

Server certificate validation for TLS

When calling any of the NEB APIs or endpoints the integrity of the TLS certificate presented can be verified using the following checks

- That the host name indicated in the certificate matches the host name of the APIs URL
- That the presented certificate is issued by one of the publicly trusted CA's listed in the documentation
- That the certificate is within its validity period
- That the signature in the certificate is valid

As an example, this can be used to verify the validity of the signing keys available from the Discovery endpoint.

Nets eID Broker signing keys

Unless otherwise specified or configured all signed tokens issued by NEB will be signed by an HSM protected key at compliance level supporting the [FIPS 140-2] level 3 or equivalent.

All publicly available certificates are found through the OpenID Connect Discovery endpoint (TLS protected).

When signing tokens, NEB will use the algorithm **ES256** (ECDSA using P-256 and SHA-256) or stronger.

Pinning signing certificates

The signing certificates used for all signed tokens will only be changed if required. This would include if the signing key is somehow compromised.

There will be support for getting upcoming signing certificate change notifications by email or by other mechanism via a bilateral agreement with NEB. It is expected, that signing certificates will be changed only if required due to security concerns.

Unless otherwise specified, the token signing certificates for will be self-signed with a very long time-to-live (10+ years).

The signing certificate for transaction tokens will be an OCES3 organization certificate, issued by the NemLog-In3 OCES3 CA. The DN of the transaction token signing certificate is available in the Environment section in this document, allowing pinning of the certificate DN.

The signing certificates will be made available through our documentation and through the agreed communication channels, allowing pinning of the specific certificates.

Supported HTTPS/TLS versions

All endpoints will require TLS 1.2 or higher.

The general guidelines and requirements for MitID Brokers will be adhered to, as a minimum and updated continuously.

JWT, JWS and JWE tokens

JWS (JSON Web Signature) and JWE (JSON Web Encryption) are the signed and encryption versions of JWT (JSON Web Token).

NEB always uses the JWS/JWE compact serialization format.

Note, that JWT tokens are always represented as either JWS or JWE.

Supported signing and encryption algorithms for JWS and JWE tokens

If not otherwise specified, the following algorithms are supported for signing- and encryption operation of JWS and JWE tokens.

The listed options here, is the complete list of supported algorithms sending JWS or JWE tokens to NEB.

Note that the signing and encryption operations follow the standards for [JWS] and [JWE].

ECDSA signatures with ES256, ES384 and ES512.

RSASSA-PKCS1-V1_5 signatures with: RS256, RS384 and RS512.

RSASSA-PSS signatures (probabilistic signature scheme with appendix) with: PS256, PS384 and PS512.

HMAC signing algorithms: HS256, HS384, or HS512

RSASSA-PKCS1-V1_5 encryption with: RSA1_5

RSAES OAEP encryption with: RSA-OAEP

ECDH-ES encryption with: ECDH-ES

Direct symmetric encryption with: A128CBC, A256CBC, A128GCM, A256GCM

Verification of tokens

ID token and Userinfo token

The basic checks are often implemented by the OIDC client library used for the integration.

- To verify the authentication response the following steps from the OIDC specification are validated: https://openid.net/specs/openid-connect-core-1_0.html#IDTokenValidation.
- The client MUST validate that the expected restrictions for **acr, ial**, **amr**, **idp** and **identitytype** are as expected.

Access- and service token

Access- and service tokens are not verified by the client application, but by receiving services who use these tokens for authorization.

Access tokens issued by NEB conforms to the JWS specification and should be validated as an JWS Oauth2 Bearer token.

- The service MUST perform standard JWS validation and the client MUST validate the expected values for the **sub** and **iss** claims.
- The service MUST validate that the **aud** claim matches the required audience for the service.
- The service MUST validate that the required **scope** claims are present in the token.

Alternatively, the service MAY call the NEB Privilege API (if the client is allowed) using the access token as authorization bearer token to receive the privileges (roles and permissions) for the end-user. In this scenario, the NEB Privilege API will perform all the required validation steps.

Transaction token

- The expected Issuer Identifier MUST exactly match the value of the iss (issuer) claim.
- The client MUST validate the signature of transaction tokens according to JWS [JWS] using the algorithm specified in the JWT **alg** Header Parameter. The client MUST use the keys provided by the Issuer (available via the Discovery endpoint).
- The client MUST validate that the signing certificate is a Voces3 certificate with the exact certificate DN specified in the environments section in this document.
- The iat Claim can be used to reject tokens that were issued too far away from the current time, limiting the amount of time that nonces need to be stored to prevent attacks. The acceptable range is client specific.
- If a nonce value was sent in the authentication request, a nonce claim MUST be present, and its value checked to verify that it is the same value as the one that was sent in the authentication request. The client SHOULD check the nonce value for replay attacks. The precise method for detecting replay attacks is client specific.
- If the **auth_time** claim was requested, either through a specific request for this claim or by using the **max_age** parameter, the client SHOULD check the **auth_time** claim value and request re-authentication if it determines too much time has elapsed since the last end-user authentication.
- The client MUST validate that the OCSP response validates for the signing certificate and that the OCSP response time is after the transaction token issued time (iat).
- The client MUST validate that the expected restrictions for **acr, ial**, **amr**, **idp** and **identitytype** are as expected.

Verification of UserInfo endpoint response

- Due to the possibility of token substitution attacks the UserInfo response is not guaranteed to be about the end-user identified by the sub (subject) element of the ID token. The sub claim in the UserInfo response MUST be verified to exactly match the sub claim in the ID token; if they do not match, the UserInfo response values MUST NOT be used.
- The client MAY introduce additional security measures by pinning the TLS certificates or by requesting a signed response.

Identity Providers

This section covers the available identity providers available via NEB.

Multiple identity providers

It is possible to specify multiple identity providers for a single user flow through NEB.

A client can be configured for multiple identity providers as a default or specify more than one identity provider in the **idp_values** request parameter.

NEB will automatically let the user select the preferred identity provider for the current flow.

As an example, setting the **idp_values** parameter to “**mitid nemid”** enables the user to login with either MitID or NemID.

Identity Provider parameters

Specific settings supported by identity providers are set by including the idp_params request parameter in the authorization request.

The idp_params parameter value is represented in an OAuth 2.0 request as UTF-8 encoded JSON (which ends up being form-url-encoded when passed as an OAuth parameter). When used in a Request Object value, the JSON is used as the value of the idp_params member.

The top-level members of the idp_params request JSON object are:

- [**idp**]: The idp in question (same value as the idp parameter).
  The available options are found in the specific identity provider section in this document.

An example idp_params request is as follows:

```
{
  "idp_params":
  {
      "mitid": {“referencetext”: “VHJhbnNmZXIgWCB0byBZ”},
      "nemid": {“remember_userid”: true, “transaction_ctx”: “Transfer X to Y”}
  }
}
```

Resulting claims

The resulting authentication flow from any identity provider will result in at least an ID token and an access token. The basic user claims are always included in the ID token while the full list of user claims is available through the UserInfo endpoint.

See the ID token section in this document for details on the basic ID token claims.

Unless otherwise stated, user claims are available through the UserInfo endpoint. In each identity provider section, additional claims included in the tokens for these providers will be explicitly stated.

MitID Demo

The MitID Demo flow emulates the MitID flow and will ask for a username and a password. Any value entered is accepted and the resulting ID token **sub** claim will be the entered username.

All clients can invoke this flow.

To enable the demo flow, set the **idp_values** request parameter to **mitid_demo**.

The resulting **acr** claim will always be set to https://broker.signaturgruppen.dk/loa/demo/0.

Supported parameters

Request parameter	**Description**
acr_values	Supported values: • https://broker.signaturgruppen.dk/loa/demo/0 • https://broker.signaturgruppen.dk/loa/demo/1 https://broker.signaturgruppen.dk/loa/demo/0 and https://broker.signaturgruppen.dk/loa/demo/1 mimics the use of the NSIS Low and Substantial levels for normal MitID flows.
idp_values	mitid_demo
identitytype_values	Supported values: • test

Scope	Description
mitid_demo	The following claims will be returned (with dummy values). • mitid_demo.username • mitid_demo.age • mitid_demo.ial_identity_assurance_level • mitid_demo.full_name • mitid_demo.transactionId • mitid_demo.referencetext • mitid_demo.referencetextheader • mitid_demo.transactiontext • mitid_demo.transactiontexttype •
ssn	Social Security Number. List of claims: • dk.cpr Will trigger MitID Demo CPR user-interaction.

ID token claims

Claim value	Possible values (dummy values)
identitytype	test
idp	mitid_demo
acr	https://broker.signaturgruppen.dk/loa/demo/0
ial	https://broker.signaturgruppen.dk/loa/demo/0

The MitID Demo uses the identity provider parameters specified for the MitID identity provider.

MitID

The MitID identity provider is the official Danish national electronic identity, replacing NemID.

More information is found here: https://digst.dk/it-loesninger/mitid/.

MitID follows the “National Standarder for Identiteters Sikringsniveauer” (NSIS) and all MitID flows is mapped to one of authentication Level of Assurance’s (LoA) found in the NSIS specification: https://digst.dk/it-loesninger/nemlog-in/det-kommende-nemlog-in/vejledninger-og-standarder/nsis-standarden/.

Level of Assurance (LoA) in MitID

A MitID flow will always result in a NSIS defined LoA value (Low, Substantial or High) set in the **acr** claim.

The default is Substantial, but this can be controlled by setting the appropriate value in the acr_values request parameter.

Setting the requested LoA to Low will allow the user to authenticate with the approved 1-factor options for MitID flows, most commonly resulting in the username + password combination. If the user chooses to select another option, the resulting LoA might be higher than Low.

Setting the requested LoA to Substantial or High will enforce a higher Authentication Assurance Level (AAL) and often result in the username + MitID App combination.

It is not possible to restrict the user’s choice of authenticators.

Supported parameters

Request parameter	**Description**
acr_values	One or more can be specified • https://data.gov.dk/concept/core/nsis/Low • https://data.gov.dk/concept/core/nsis/Substantial (default) • https://data.gov.dk/concept/core/nsis/High
idp_values	**mitid**
ial_values	One or more can be specified • https://data.gov.dk/concept/core/nsis/Low • https://data.gov.dk/concept/core/nsis/Substantial (default) • https://data.gov.dk/concept/core/nsis/High

Supported identity provider parameters

Identity Provider parameters (mitid)	**Description**
referencetext	Type: Base64 encoded string. The reference text containing the transaction content (e.g. “Transfer <amount> to <ac-count>”). This text will be displayed to the user in all MitID flows inside the MitID client. It will be shown to the user in the MitID app. It is limited to 130 characters.
referencetextheader	Type: Base64 encoded string A short header for the referencetext
transactiontext	Type: JSON. The transaction text is presented to the end-user as part of the MitID flow and allows service providers to provide a transactional context for the MitID flow. The top-level members of the **transactiontext** request JSON object are: • value: The text or html to display in UTF-8+Base64 encoding. • type: One of [**text**, **html**]. The **transactiontext** parameter value is represented as UTF-8 encoded JSON (which ends up being form-url-encoded when passed as a request parameter).
transactiontexttype	Type: String

	One of <ul><li>text</li><li>html</li></ul>If text is specified, the referencetext will displayed “as-is” without any rendering, as a plain text value. If html is specified, the content will be displayed and rendered as HTML. The allowed HTML is restricted as specified in this document.

Scope	**Description**
mitid	List of claims: <ul><li>mitid.uuid</li><li>mitid.date_of_birth</li><li>mitid.age</li><li>mitid.cpr_name</li><li>mitid.ial_identity_assurance_level</li><li>mitid.riskdata</li><li>mitid.referencetext</li><li>mitid.referencetextheader</li></ul>
ssn	Social Security Number. List of claims: <ul><li>dk.cpr</li></ul>Will trigger MitID CPR user-interaction.

ID Token identity claims

Claim value	**Possible values**
identitytype	private
idp	**mitid**
acr	One of <ul><li>https://data.gov.dk/concept/core/nsis/Low</li><li>https://data.gov.dk/concept/core/nsis/Substantial</li><li>https://data.gov.dk/concept/core/nsis/High</li></ul>
ial	One of <ul><li>https://data.gov.dk/concept/core/nsis/Low</li><li>https://data.gov.dk/concept/core/nsis/Substantial</li><li>https://data.gov.dk/concept/core/nsis/High</li></ul>
amr	The list of authenticators used to achieve the resulting LoA. The list is a space separated list of string values. Possible values are: <ul><li>mitid.password</li><li>mitid.code_token</li><li>mitid.code_reader</li><li>mitid.code_app</li></ul>

	• mitid.code_app_enchanced • mitid.u2f_token

Transaction token MitID specific claims

Claim value	**Possible values**
mitid.uuid	Same value as for ID token.
mitid.referencetext	Passthrough of the MitID **referencetext** identity provider parameter.
mitid.transactiontext	Passthrough of the MitID **transactiontext** identity provider parameter.
dk.cpr	Danish CPR. Included if and only if configured as a requirement for the transaction token for the client in the administration interface. Will trigger the same user interaction as setting the **ssn** scope.

MitID CPR flow

NOTE: The way CPR is handled in MitID and by NEB is subject to change.

CPR is available from MitID flows if you are a public service provider. In this scenario, NEB will set **dk.cpr** in the result, if requested via the **ssn** scope.

If you are a private service provider, the user's CPR will not be available from the MitID system. In this scenario, a CPR Match service is provided (see MitID CPR Match API), available for MitID Brokers making it possible to match an active MitID session and CPR and verify if the supplied CPR matches the authenticated MitID identity and thus making it possible to verify if a MitID identity has the given CPR.xxxxxx

NEB implements this as a natural part of the MitID flow and will ask the user for CPR when the service provider requests CPR with the **ssn** scope.

The end-user will have the option (consent) to have the CPR stored for later use for the same service provider and thus allows the end-user to avoid having to enter the CPR when authenticating for the same service provider again.

*Note, that it is supported to request CPR via the CPR flow by reauthenticating a user with the additional **ssn** scope. In this case, NEB will reuse the active MitID session and ask the user for CPR (but not ask for login), do the required CPR Match verification, and return the CPR to the service. This enables services to only ask for CPR using the CPR flow when needed for specific users.*

MitID CPR Match API

NOTE: The way CPR is handled in MitID and by NEB is subject to change.

The Broker API supports a "MitID CPR Match API" that allows services to match a CPR with a MitID authentication from NEB.

In this way, services can ask the user for CPR and then call the API with the access token retrieved from NEB for the user authentication as authorization header.

This also allows services to verify that an already known CPR matches the MitID identity in question.

NemLog-In3

NOTE: Subject to change. Integration not initiated yet.

https://en.digst.dk/digitisation/nemlog-in/

NemLog-in is a common log-on solution which gives access to the public authority self-service solutions both in the municipalities, regions, and the government.

With NemLog-in you only need to log on once to identify yourself to all the various public authority self-service solutions. With your NemLog-in, you have access to many different service providers and public services.

Request parameter	**Description**
acr_values	Possible values • https://data.gov.dk/concept/core/nsis/Low • https://data.gov.dk/concept/core/nsis/Substantial (default) • https://data.gov.dk/concept/core/nsis/High
idp_values	**nemlogin**
identitytype_values	Possible values • private • professional

Scope	**Description**
nemlogin	List of claims: • nemlogin.ial • nemlogin.aal • nemlogin.name • nemlogin.given_name • nemlogin.family_name • nemlogin.email Only for professional identities: • nemlogin.auth_to_repr • nemlogin.p_number • nemlogin.se_number • nemlogin.persistent_id • nemlogin.cvr • nemlogin.org_name
ssn	Social Security Number. List of claims: • dk.cpr • dk.cpr_uuid
nemlogin.priviledges	List of claims: • nemlogin.priv Only Danish public service providers can request this scope.
organization	• organization.number • organization.name

ID Token identity claims

Claim value	Possible values
identitytype	One of • private • professional Private identities are mapped from the NemLog-In person identity type. NemLog-In specifies **person** and **professional** as identity types.
idp	**nemlogin**
acr	One of • https://data.gov.dk/concept/core/nsis/Low • https://data.gov.dk/concept/core/nsis/Substantial • https://data.gov.dk/concept/core/nsis/High
ial	One of • https://data.gov.dk/concept/core/nsis/Low • https://data.gov.dk/concept/core/nsis/Substantial • https://data.gov.dk/concept/core/nsis/High
amr	Not specified yet.

NemID

The NemID identity provider is the official Danish national electronic identity, being replaced by MitID.

Supported parameters

Request parameter	Description
acr_values	• https://data.gov.dk/concept/core/nsis/Substantial
ial_values	• https://data.gov.dk/concept/core/nsis/Substantial
idp_values	**nemid**
amr_values	The following options are supported • nemid.otp (default) • nemid.keyfile If all options are set, the user will be able to choose between the two NemID client variants: NemID OTP and NemID key file. NemID OTP is mostly used for private identities. NemID key file is mostly used for professional identities.

Supported identity provider parameters

Identity Provider parameters (nemid)	Description

remember_userid	Type: bool (default: false) If true, the user is shown the option to “remember me” in the NemID OTP client.
code_app_trans_ctx	Type: string. Up to 100 characters. Showed in the NemID Code App if the end-user choses the NemID Code App when authenticating.
sign_text	Type: JSON. NemID Sign Text. If set, invokes the NemID Sign flow. The **sign_text** parameter has the following top-level members • **value** • **type** The value member is the UTF-8 + Base64 encoded signtext. The type member is a string with one of the following values • text • pdf

Scope	**Description**
nemid	User claims for NemID, that does not require consent. List of claims: • **nemid.common_name** • **nemid.pid** • **nemid.rid** • **nemid.dn** • **nemid.ssn** • **nemid.email** • **nemid.code_app_trans_ctx** • **nemid.sign_text** • **nemid.xmldsig** • **nemid.cvr**
ssn	Social Security Number. List of claims: • dk.cpr

ID Token identity claims

Claim value	**Possible values**
identitytype	• private • professional

	Private identities are identifiable by their global NemID PID, found in the **nemid.pid** claim. Professionals are employees, and have a unique NemID RID, found in the **nemid.rid** claim. The NemID RID paired with the CVR from the employer forms the primary identifier for NemID professional identities.
idp	**nemid**
acr	• https://data.gov.dk/concept/core/nsis/Substantial
ial	• https://data.gov.dk/concept/core/nsis/Substantial
amr	One of the following: • nemid.otp • nemid.keyfile

Transaction token NemID specific claims

Claim value	**Possible values**
nemid.ssn	Same value as for ID token.
nemid.code_app_trans_ctx	Passthrough of the NemID **code_app_trans_ctx** identity provider parameter.
nemid.sign_text	Passthrough of the NemID **sign_text** identity provider parameter.
dk.cpr	Danish CPR. Included if and only if configured as a requirement for the transaction token for the client in the administration interface. Will trigger the same user interaction as setting the **ssn** scope.

NemID CPR

In the current version of NemID, CPR is available from NemID flows if you are a public service provider. In this scenario, NEB will set **dk.cpr** in the result, if requested via the **ssn** scope.

If you are a private service provider, the user's CPR will not be available from the MitID system. In this scenario, a CPR Match service is provided (using the service providers NemID agreement) making it possible to match a NemID PID and CPR and verify if the supplied PID and CPR matches and thus making it possible to verify if a NemID identity has the given CPR.

NEB implements this as a natural part of the MitID flow and will ask the user for CPR when the service provider requests CPR with the **ssn** scope.

If the user has accepted, that the CPR is stored for later use (user consent) and returned to the service provider, the user will not have to enter CPR for subsequent MitID flows for the same service provider.

NemID CPR Match API

The Broker API supports a "NemID CPR Match API" that allows services to match a CPR with a NemID authentication from the NEB.

In this way, services can ask the user for CPR and then call the API with the access token retrieved from NEB for the user authentication as authorization header.

This also allows services to verify that an already known CPR matches the NemID identity in question.

NemID Privat til Erhverv (authorized to represent)

The “NemID Privat til Erhverv” service is available via NEB using the **nemid.poces_to_cvr** scope (client must be allowed to do so).

This allows the end-user to use his Private NemID to represent a company of which he or she is in full control. NEB will handle all the end-user interaction and return the selected company identifier in the **nemid.auth_to_repr** claim.

References

1. [OIDC] “OpenID Connect core”: https://openid.net/specs/openid-connect-core-1_0.html
2. [OIDC-DISC] “OpenID Connect Discovery”: https://openid.net/specs/openid-connect-discovery-1_0.html
3. [ROBJ] “Passing Request Parameters as JWTs”: https://openid.net/specs/openid-connect-core-1_0.html#JWTRequests
4. [JWT] “JWT specification”: https://tools.ietf.org/html/rfc7519
5. [JWS] “JWS specification”: https://tools.ietf.org/html/rfc7515
6. [JWE] “JWE specification”: https://tools.ietf.org/html/rfc7516
7. [JWA] ”JWA specification”: https://tools.ietf.org/html/rfc7518
8. [NSIS] ”National Standard for Identiteters Sikringsniveauer 2.0.1”: https://digst.dk/it-loesninger/nemlog-in/det-kommende-nemlog-in/vejledninger-og-standarder/nsis-standarden/
9. [OAuth] “The OAuth 2.0 Authorization Framework”: https://tools.ietf.org/html/rfc6749
10. [OAuth Native] “OAuth 2.0 for Native Apps”: https://tools.ietf.org/html/rfc8252
11. [Chrome Ext Tabs] “Chrome custom tabs”: https://developer.chrome.com/multidevice/android/customtabs
12. [PKCE] “Proof Key for Code Exchange”: https://tools.ietf.org/html/rfc7636
13. [JWT JWS JWE] “JWT, JWS and JWE”: https://medium.facilelogin.com/jwt-jws-and-jwe-for-not-so-dummies-b63310d201a3
14. [OIO PRIV] ”OIO Basic Privilege Profile”: https://digst.dk/media/20999/oiosaml-basic-privilege-profile-1_2.pdf
15. [OIOSAML] ”OIOSAML 3.0.1”: https://digst.dk/media/21892/oiosaml-web-sso-profile-301.pdf