SignaturGruppen

Technical reference

for service providers

Version 0.9.7.3 2020

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-standard / 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.

Changelog

Version 0.9.7.3 - 28/1-2021

- Added reference to Oauth 2.0 Token Exchange reference: [TOKEN-EXCHANGE]
- Updated description of MitID SSO and MitID Controlled Transfer in the MitID identity provider section.
- Updated description of SSO and SLO.
- Added definition and description of Logout endpoint.
- Updated ID Token MitID specific claims and Transaction Token MitID specific claims to include mitid.psd2 claim.
- Updated description of NemID Private to Business and added description of the nemid.auth_to_repr claim.
- Added description of MitID Controlled Transfer Token Exchange Code Api endpoint.

Version 0.9.7.2

- Added note about Administration API available not before after MitID GO-Live.
- Added transasction_claims scope description for MitID.
- Added transasction_claims scope description for MitID Demo.
- Added transasction_claims scope description for NemID.

Version 0.9.7.1 - 12/12 -2020

- Added the “SSO” section
- Added the “Sessions and session management” section
- Added MitID SSO and MitID Controlled Transfer description in the MitID section.
- Removed amr_values request parameters for MitID Demo.
- Removed ial_values as a request parameter in general. Might be reintroduced later.
- Removed amr_values as a general request parameter (might be introduced later again).
- Removed amr_values request parameters from NemID.
- Removed amr_values request parameters from BankID SE.
- Removed dk.cpr claim from MitID transaction token
- Removed transactiontexthheader as parameter from MitID flows
- Added MitID uuid_hint idp parameter description.
- Updated amr values for BankID SE idp.
- Removed prompt=select_account as option. Might be reintroduced at later stage.
- Removed identity_type_values (old name: identitytype_values) as a request parameter. Will be reintroduced when NL3 is made available but might be as idp specific parameters.

Version 0.9.7.0

- Added this changelog.
- Updated introduction disclaimer about “work-in-progress”.
- General text/wording updates.
- Updated description of Signed requests.
- Updated description of “Administration web-interface and API”.
- Updated description of Step-up handling
- Added small example of user-flow error codes.
- Updated certificate info for pilot-test environment
- Updated Broker API section with more detailed examples and better description.
- Updated issued token descriptions
- Updated description of the **prompt** authentication parameter.
- Added the [OIDC-SESSION] reference.
- Added the Custom API resources section.
- Updated description of reauthentication with information about max age and auth_time claim.

- Updated scopes description.
- Added “Privat NemID/MitID to Erhverv” updates for the NemID and MitID identity provider sections.
- Added a small subsection under NemLog-In3 identity provider: Signature at NemLog-In3.
- Added consent for SSN storage available for service providers for MitID, NemID and NL3 identity providers.
- Added small section about “Custome identity providers”
- Rename to “National Danish identity providers”
- Added “Nets E-Ident identity providers” section.
- Added description of BankID Norway
- Updated Identity providers sections and general descriptions.
- Added a more generic description of supported E-Ident service providers.

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.

Note that many features described in this document are under development.

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

All administration and configuration for service providers is handled through the **Administration web-interface (ADM-UI)**, which allows Nets eID Broker or service provider administrators to configure 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.

It will also be possible to control and configure branding, layout, and default settings for various parameters.

ADM-UI also supports privilege administration of NEB privileges, as well as administration of own privileges.

[Note: Not currently available for service providers. The ADM-UI will be made available and described in its own documentation when ready for external customer access. No date has been set for external release yet.]

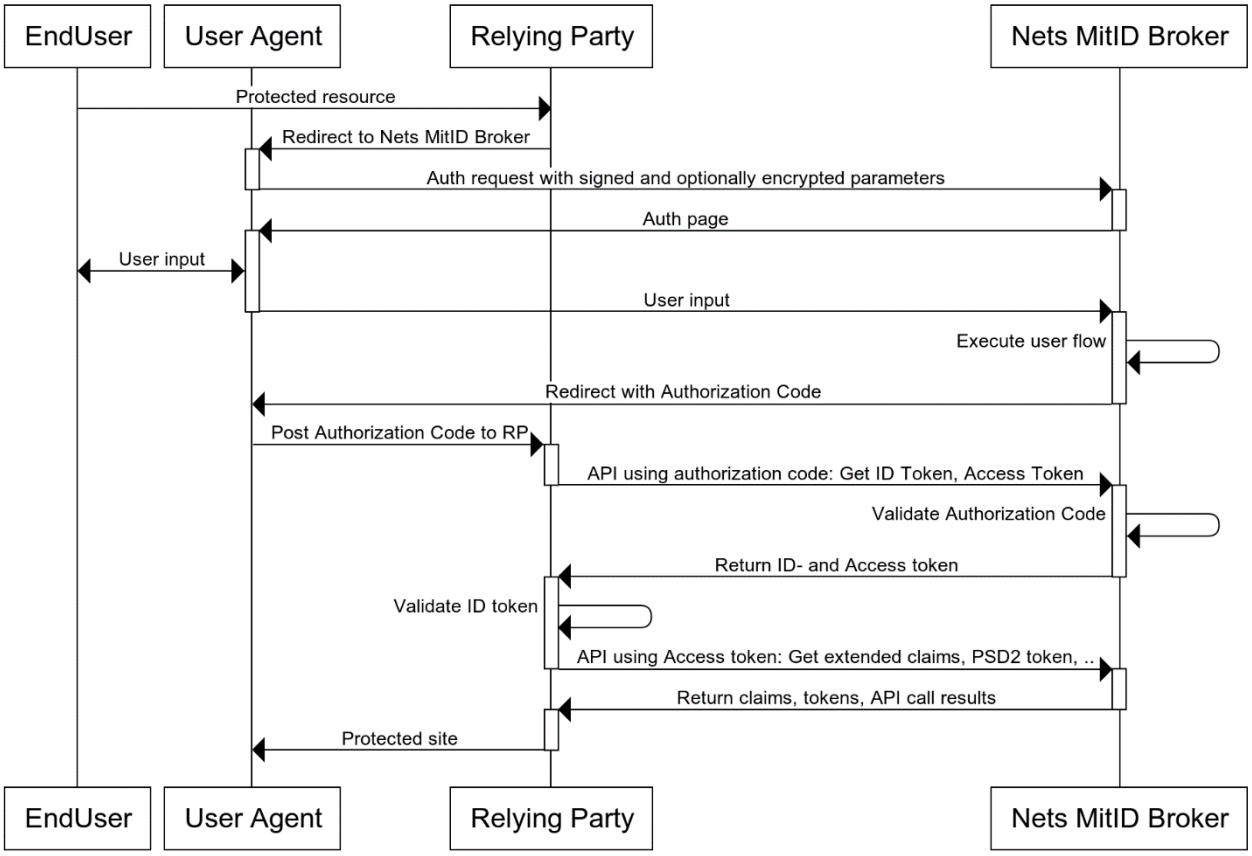
Web and app client integration

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.



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 determine 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 that 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 section 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 authentication. 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 possible, 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.

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.

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. Note that **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. Supported values are • **none** (default) • **login** (force authentication request) If **login** is used, the end-user will be forced to complete the requested authentication flow. This can be used anytime it is required that the end-user completes the requested authentication flow, i.e. when requesting a signature from the end-user.

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.
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.

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 enter a step-up flow. This is automatically handled by NEB based on the active session and the requested authentication.

An example of a step-up flow would be sending the user for authentication to a higher NSIS Level of Assurance for a MitID authentication, 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 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.

Max age and authentication time

The **max_age** authentication parameter can be used to control how old an existing session can be before triggering a new authentication entirely.

The **auth_time** claim in the ID token will always contain the authentication time of the session which the token is issued from. In this way the **auth_time** can be verified to ensure that the authentication was processed within the expected and/or allowed timeframe.

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 supports 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.

A full list of error codes and descriptions will be made available in this document. Error codes have not been thoroughly defined yet.

Examples of error codes:

OP006	User aborted
OP007	Unable to find valid IDP for authentication

Example response:

```
client_redirect_uri?error=OP006&state=xyz
```

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.

Examples are the identity provider specific scopes that maps to the list of identity provider specific user-claims issued in the flow, e.g. if the **mitid** scope is specified all non-empty mitid.* claims will be issued.

All requested scopes are mapped directly to the scope claim in the issued access- and service tokens, allowing scopes to be used in authorization contexts.

An API Resource is a scope that maps to one or more API's for which it is intended. These API's are represented by their audience, are mapped to the **aud** claim of the issued access- or service tokens. See the description of Custom API resources later in this document.

Some scopes trigger user-interaction or user-consents as they might map to claims which requires special actions or consents to be issued.

If a client is requesting a scope which is not allowed for this client, the entire flow will fail.

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 – https://brokertest.signaturgruppen.dk/op

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

Token signing certificates (self-signed)

Subject	CN = Nets eID Broker - AzureTest Signing Credential
Subject Thumbprint (KID)	27d04dff8c339dcab0121c0a6fec6b9206a310c7

SSL certificates

Subject	CN = *.signaturgruppen.dk
Subject Thumbprint (KID)	21ca342d9d98c3d403f5cf2a4bc8d68a2518404e
CA Subject	CN = DigiCert Global Root CAOU = www.digicert.comO = DigiCert IncC = US
CA Thumbprint	a8985d3a65e5e5c4b2d7d66d40c6dd2fb19c5436

Transaction token signing cert

Subject	CN = SIGNATURGRUPPEN A/S - Nets eID Broker Test
	SERIALNUMBER = CVR:29915938-UID:51799026
	O = SIGNATURGRUPPEN A/S // CVR:29915938
	C = DK

Subject Thumbprint (KID)	78f1c9fdb9db662ed04905c847cd87a87d7e7a57
CA Subject	CN = TRUST2408 Systemtest XXXIV CA O = TRUST2408 C = DK
CA Thumbprint	eeaf09230cd54e31a22872bd83cd189095921ad7

Nets eID Broker API

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

The available 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. Will be described in its own document at a later stage.
- **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]. Will be described in its own document at a later stage.

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 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 do not 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 URLs 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.

All Broker API endpoints have their endpoints published via the Discovery endpoint.

The Discovery endpoint will dynamically list available endpoints, available scopes and various other relevant OIDC specific information.

Token endpoint

All tokens issued by the NEB platform is issued from the Token endpoint. The endpoint follows the OpenID Connect specification.

UserInfo endpoint

For most user authentication flows, the resulting access token provides access to the UserInfo endpoint by using the access token as an authentication bearer token.

The endpoint returns the full list of issued user-claims for the end-user, if the end-user session is active at NEB.

The result from the Userinfo endpoint may contain the following information

- User-claims from the associated end-user session.
- Session info from the associated end-user session.
- Transaction specific claims, like signing texts – bound to the specific access token. This information is available 1 hour from issue time.

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”                          : “35”,
  “mitid.date_of_birth”                : “1985-03-29”,
  “mitid.ial_identity_assurance_level” : “SUBSTANTIAL”,
  “da.cpr”                             : “290385-xxxx”,
  “mitid.cpr_name”                     : “Hans Hansen”,
  “session_status”                     : “active”,
  “session_identifier”                 : “19712D37-0C76-4AAA-B049-BACD585F484F”,
  “session_acr”                        : “https://data.gov.dk/concept/core/nsis/Substantial”,
```

}

Logout endpoint

Under review, not available yet.

The Logout endpoint is a NEB specific endpoint performing the SLO ceremony involved for logging out the end-user session.

The endpoint is exposed in the discovery document with the following definition:

```
POST {Authority URL}/v1/sso/logout
 Authorization: Bearer [access token]
 Content-Type: application/x-www-form-urlencoded
```

Required scope	Description
sso_api	It is required, that the client has requested the mitid_api scope to be allowed to invoke the MitID Controlled Transfer Token Exchange Code API.

NOTE: The logout endpoint is only available for logging out back-channel SSO group sessions.

Administration API

The administration API will become available at some point after MitID GO-Live.

The administration API supports the full functionality available through our administrative interface. This will be documented in a separate document which will be accompanied with a OpenAPI 3.0 specification (Swagger).

The NEB platform will provide a web interface, but access to the API can also be done directly by own integration.

When using the API directly, accessing the API with a client which has been granted access to request services tokens (Client Credentials Grant) with appropriate privileges is required.

Privileges and Privilege API

The Privilege API will become available with NemLog-in3 professional identities, currently scheduled for GO-Live December 2021.

The NEB platform defines a Privilege API to support an extended model and usage of roles and permissions which is compatible with both the Danish “OIOSAML Basic Privilege Profile” [OIO PRIV] and the Nets Attribute Service.

The administration and assignment of privileges will be available through the Administrative API and web-interface.

The scope **privileges** can be requested when making authentication requests, to request the **privileges** claim issued via the Userinfo endpoint, which will contain the privileges assigned to the authenticated identity which is available for the receiving service.

The Privilege API (not yet defined) will support privilege requests and require an NEB issued access- or service token as authorization bearer token.

The full documentation for the Privilege API and usage of privileges in the NEB platform will be described in a separate document.

Single-Sign-On and Single-Log-Out (SSO and SLO)

Note that in the current pilot-environment a single one-identity session is active. The described functionality in this section has not yet been activated.

The NEB platform defines its own SSO capabilities, as well as supporting MitID SSO and MitID Controlled Transfer.

This section describes how Single-sign-on (SSO), Single-log-out (SLO) and sessions are handled in the NEB platform.

MitID SSO and MitID Controlled Transfer functionalities are described in the MitID identity provider section later in this document.

SSO groups

SSO groups define the scope of end-user sessions and define the scope and context of which each authentication is taking place.

All services are members of exactly one SSO group, which then defines the scope for the user-session and defines the behavior for SSO and SLO.

A service can be in an SSO-group on its own or being part of an SSO group with any number of other services.

And end-user will share all sessions between all services joined in the same SSO group which enables SSO and SLO between these services.

SSO groups can support either back-channel logout only or both front-channel and back-channel logout.

SSO groups that support both front-channel and back-channel logout is called a Front-channel SSO group and require all participants who want to be informed about session logout to support either a front-channel or a back-channel logout endpoint.

SSO groups that support only back-channel logout is called a Back-channel SSO group and requires all participants who wants to be informed about session logout to support a back-channel logout endpoint.

Example:

If a service does not want to share session state with other services, the SSO group containing only the service itself will enable this. Sign-in and log-out will affect only the sessions created for the service and will affect no other services.

SSO groups can cover a single service, a service provider, an organization, or a collaboration between multiple organizations and some of their services.

SLO

Single-log-out will trigger log-out of the specified user-session and will trigger a federated log-out by calling all relevant clients (services) and return the end-user to desired location.

SLO for Front-channel SSO groups

A SLO can be performed by sending the end-user to the NEB End session endpoint with the relevant ID token issued to the client representing the user-session. The following will take place for each client that has received a log-in from this session:

- If required, a log-out of the upstream identity provider is executed.
- If the client has registered a front-channel log-out endpoint, this will be invoked using and iframe in the end-user's browser.
- If the client has registered a back-channel log-out endpoint, this will be called by the NEB backend.

When all clients have been processed, the end-user will be returned to the calling service if allowed and requested.

For more information on the Front-Channel and the Back-Channel OpenID Connect specifications, please refer to [OIDC-FRONT-CHANNEL] and [OIDC-BACK-CHANNEL].

SLO for Back-channel SSO groups

SLO can be performed by sending the end-user to the End session endpoint as for Front-channel SSO groups but can also be performed by calling the Logout endpoint, which is described in more detail in the Nets eID Broker API section.

The Logout endpoint is an API endpoint called with the originally issued ID token which then performs the logout ceremony required to logout the end-user. This requires all participants to support Back-channel logout.

Sessions and session management

Sessions

When the end-user authenticates, NEB creates and maintains a user session holding the issued claims and required information. If the session is active the relevant user claims are accessible via the Userinfo endpoint using the issued access token.

The authenticated session is shared between all authenticated members of the SSO group.

Session management

Session management is supported by the methods described in this section.

OpenID Connect session management

OpenID Connect session management is not supported yet.

The session management specification [OIDC-SESSION] enables services to check if the user session has changed using a hidden iframe. This enables a pure browser-based mechanism for verifying if the end-user session has changes since the services received the authentication response and is done entirely using JavaScript, an iframe and a special cookie issued at the NEB domain.

NOTE: OIDC session management is only supported in Front-channel SSO Groups.

OpenID Connect Prompt none

An authentication request with the **prompt=none** will return with a specific error code if the end-user session is no longer valid for the requested authentication or return with updated tokens if the session is still active.

NEB Userinfo endpoint using access token

Not supported yet.

The NEB Userinfo endpoint will return the following session specific claims when the **session_info** scope is requested in the authorization request.

Claim	Value
session_status	Possible values • active • inactive Indicates the session state. If inactive is returned, then the user session no longer exists. If active is returned the session is still valid. The rest of the claims in this table is only returned if the session_state is active.
session_identifier	The Session ID – also found in the **sid** claim in the ID token. The associated session identifier.

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.
- **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.
- **Service token (Client Credentials Grant)**: Token issued directly to a service.

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 userinfo token contains most of the ID token claims and all the claims issued by the Userinfo endpoint. This provides a signed format for all user-claims.

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 a 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.

The transaction token should only be requested if required for internal revision or similar requirements.

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 are issued from the Token endpoint or via the browser in Hybrid or Implicit flows.

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.
sub	NEB specific UUID representing the authenticated end-user. Primary end-user identifier. Always the same for the same identity, i.e. the same MitID identity will have the same sub claim.
aud	Audience(s) that this ID Token is intended for. This will be the ClientID of the receiving party.
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.
identity_type	One of • private • professional • test
transaction_id	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",
 "identity_type"                      : "private",
 "idp"                                : “mitid”,
 "iat"                                : 1311290550,
 "exp"                                : 1311291550,
```

```
 "...."                          : ....,
 "spec_ver"                      : "0.9"
}
```

Default expiry for ID tokens is 5 minutes.

The ID token is used when log-out is called for the end-user or when (re-)authenticating the end-user and forcing the same end-user to complete the transaction. The ID token is designed to include only the minimal required claim values to work as both browser-issued token in some scenarios, as well as logout token.

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 Userinfo endpoint or the Privilege API.
- Access internal APIs (i.e. internal to the Organization/Service in question)
- Access external APIs

Default expiry for access tokens is 1 hour.

Service token (Client Credentials Grant)

If allowed, a service can retrieve a service token from the Token endpoint.

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

Service tokens are used as bearer tokens exactly as access tokens. Service tokens are issued directly to a client (service) where an access token is issued to a client on behalf of an end-user.

Service tokens can have privileges and scopes just as access tokens, which defines the privileges and API's that the service has been granted access to.

Service tokens are issued by using the Client Credentials Grant (section 4.4. in [OAuth]).

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.

Usage and format follow from the [OIDC] specification.

The intended usage of Refresh tokens is to enable long-lived sessions for the end-user in the target application, typically a mobile App.

Refresh tokens can be used to retrieve access tokens over a long period of time and by this mechanism enable an application to maintain access to specified API's during the lifetime of the issued Refresh token.

Refresh tokens can be revoked, enabling a mechanism to restrict further usage of a specific Refresh token.

Userinfo token

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

This allows the retrieval of all user claims directly from the Token endpoint in a signed format, signed with the same signing key as the ID- and access token.

The Userinfo token is requested by setting the scope value **userinfo_token** or can be set to always be returned for a specific client.

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- or Userinfo 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 on 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, if supported by the OCES3 setup.

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.
identity_type	One of • private • professional • test
transaction_id	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.

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 **identity_type** 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 **identity_type** 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.

Custom API resources

Enterprise feature. Contact Signaturgruppen for details and expected time for going live.

Custom API resources can be created which will define access to one or more APIs and will allow the usage of NEB issued access- or service tokens to be authorized for the specified APIs.

An API resource defines a scope values which a client can utilize to request authorization for the APIs defined for the API resource.

An API resource consists of.

- A unique scope value name
- A list of claims-types to add to the token response
- A list of audiences (e.g., https://api.example.com)

The resulting access- or service token will include the requested scope values and will include the audience and claim values specified for each requested API resource.

Example:

The API at https://api.example.com will be able to define the API resource:

- Scope: api-aannd8 (auto generated by NEB)
- Audience https://api.example.com

When a client requests the scope api-aannd8 and has permission to do so, the resulting access- or service tokens include the scope api-aannd8 and audience https://api.example.com.

This allows APIs at https://api.example.com to utilize the NEB issued access- and service tokens as authorization bearer tokens.

Identity Providers

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": {“reference_text”: “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.

Custom identity providers

It will be possible to setup integration to custom identity providers supporting OpenID Connect or SAML/WS-Federation based integrations.

This will allow NEB to handle authentications towards ex. Microsoft Azure, AD FS based setups, or any other internal or external identity provider based on either OpenID Connect or SAML.

Custom identity providers are handled like any other identity provider by NEB and allows integration with NEB SSO and other enterprise features.

Danish identity providers

This section covers the available national identity providers available via NEB.

Supported Danish identity providers

- MitID Demo
- MitID
- NemID
- NemLog-In3

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
idp_values	mitid_demo

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
transasction_claims	• mitid_demo.reference_text • mitid_demo.transaction_text • mitid_demo.transaction_text_type • mitid_demo.transaction_id
ssn	Social Security Number. List of claims: • dk.cpr Will trigger MitID Demo CPR user-interaction.
ssn_store	Request permission to store the SSN requested with the ssn scope. If this scope is specified, the user will be given the option to allow the requesting service provider to store the end user SSN, ex the Danish CPR number. This will be shown to the end-user as a checkbox with the appropriate text. When requested the Userinfo endpoint will issue the following claim type: • ssn_store_consent The ssn_store_consent claim value can be one of • “allowed” • “denied” (default)

ID token claims

Claim value	Possible values (dummy values)

identity_type	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/.

Supported OIDC 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**

Supported identity provider parameters (idp_params -> mitid)

Identity Provider parameters (mitid)	Description
reference_text	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.
transaction_text	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 **transaction_text** request JSON object are:

	• value: The text or html to display in UTF-8+Base64 encoding. • type: One of [**text**, **html**]. The **transaction_text** parameter value is represented as UTF-8 encoded JSON (which ends up being form-url-encoded when passed as a request parameter).
transaction_text_type	Type: String One of • text • html If text is specified, the reference_text will be 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.
uuid_hint	Type: String If this parameter is set with the end-user MitID UUID, the MitID flow will be automatically started for the MitID identity with the specified MitID UUID.
psd2	Type: Boolean If this parameter is set to true, the individual authentication flow will be PSD2 compliant by restricting what information can be revealed to the end-user during authentication.

Scope	**Description**
mitid	List of claims: • mitid.uuid • mitid.date_of_birth • mitid.age • mitid.cpr_name • mitid.ial_identity_assurance_level
transaction_claims transaction_token	The transaction_claims scope maps the following claims to the Userinfo endpoint output. The claims will only be available from the Userinfo endpoint with the specific access token issued from the end-user authentication mapping the relevant claims. The transaction_token scope requests the transaction token from the Token endpoint including the following claims. These claims are not shared in a SSO with other services. • transaction_id • mitid.transaction_text • mitid.transaction_text_type • mitid.reference_text • mitid.transaction_id
ssn	Social Security Number.

	List of claims: • dk.cpr Will trigger MitID CPR user-interaction.
ssn_store	Request permission to store the SSN requested with the ssn scope. If this scope is specified, the user will be given the option to allow the requesting service provider to store the end user SSN, i.e. the Danish CPR number. This will be shown to the end-user as a checkbox with the appropriate text. When requested the Userinfo endpoint will issue the following claim type: • **ssn_store_consent** The ssn_store_consent claim value can be one of • "allowed" • "denied" (default)

ID Token identity claims

Claim value	**Possible values**
identity_type	private
idp	**mitid**
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	The list of authenticators used to achieve the resulting LoA. The list is a space separated list of string values. Possible values are: • mitid.password • mitid.code_token • mitid.code_reader • mitid.code_app • mitid.code_app_enchanced • mitid.u2f_token
mitid.psd2	The mitid.psd2 claim is only issued as an ID Token identity claim if the authentication of an end-user is PSD2 compliant.

Transaction token MitID specific claims

Claim value	**Possible values**
mitid.uuid	Same value as for ID token.
mitid.reference_text	Passthrough of the MitID **reference_text** identity provider parameter.

mitid.transaction_text	Passthrough of the MitID **transactiontext** identity provider parameter.
mitid.transaction_text_type	Passthrough of the MitID **transactiontexttype** identity provider parameter
mitid.psd2	The mitid.psd2 claim is always issued as a transaction token MitID specific claim.

MitiD Transaction signing

Nets eID Broker supports a transaction signing flow which enables the end-user to approve a transaction text based on text or HTML, as part of the MitID authentication.

This is done by setting the **transaction_text** and **transaction_text_type** MitID identity provider parameters.

The end-user will be shown the text/HTML and will have to approve the text to complete the transaction.

MitID natively supports the **reference_text** (130 characters) parameter which enables a limited size and format to present the end-user with detailed information about the transaction.

If the transaction token is requested, a NEB sealed record of the transaction is returned including all the relevant parameters used to complete the transaction.

MitID CPR flow

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.

If the **ssn_store** scope is requested, the end user will be able give consent allowing the service provider to store the ssn.

*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 will 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

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.

MitID Privat til Erhverv (authorized to represent)

The "MitID Privat til Erhverv" service is available via NEB for MitID flows.

The specific parameters have not yet been defined, but it will allow the end-user to select between the available CVR-numbers available for which his or hers private MitID is "Allowed to Represent" the specified company.

This requires that the user enters his CPR number (or that the calling service is allowed for automatic CPR retrieval) after which the CVR-list can be retrieved from the Danish “Erhvervsstyrelsen” and the flow can be completed.

MitID SSO

NEB implements and supports MitID SSO and allows integrating services to utilize MitID SSO for automatic sharing MitID sessions in a SSO defined and managed by participating MitID Brokers.

Any client, service or service provider can be member of up to one MitID SSO Group and if so, will automatically map all MitID sessions to this MitID SSO and automatically reuse an existing session if already active within this MitID SSO.

It is possible to setup MitID SSO groups with other MitID Brokers and other MitID Broker services and service providers.

NEB will handle the required end-user consent, which is required when automatically reusing an active MitID session.

MitID SSO logout

It is required by all clients who utilize a MitID SSO to inform their MitID broker of logout events from the end-user.

Logout is handled either by sending the end-user to the End session endpoint with the original issued ID token or by calling the Logout endpoint with the original issued ID token. See general section about logout in this document for more details.

MitID SSO requires that all logout events are handled using Back-channel endpoints and thus enabling the termination of MitID SSO sessions without the need of the end-user browser. MitID SSO groups are by this forced to be Back-channel SSO groups.

The only way participating service providers can receive logout events is by registering a valid Back-channel endpoint for their integration at NEB.

Participating service providers will be able to get the session status from the Userinfo endpoint.

MitID Controlled Transfer

MitID Controlled Transfer (MitID CT) implementation and API is under development. The API is not available yet, but the draft specification is available here.

With MitID Controlled Transfer, a requesting service provider can request and retrieve a “MitID Controlled Transfer Token Exchange Code” (MitID CT Exchange Code) from their MitID Broker. Then, another service provider can exchange this to a MitID authentication from their respective MitID Broker.

Flow:

- Service provider A requests a MitID CT Exchange Code from Broker A and specifies a Transfer Token Text
- Service Provider A redirects end-user to Service Provider B with the MitID CT Exchange Code and Transfer Token Text
- Service Provider B uses Broker B and the implementation provided by Broker B to exchange the MitID CT Exchange Code token and the Transfer Token Text to a MitID authentication enabling the end-user login at Service Provider B.

The protocol for exchanging MitID CT Exchange Code between service providers are up to the two exchanging service providers to define.

The protocol for retrieving or exchanging MitID CT Exchange Code between service providers and MitID brokers are up to each broker to define.

The specification for retrieval and exchange towards the NEB interface is specified here. Note that it is up to each agreement with other service providers, how the MitID CT Exchange Code is exchanged – this is not specified nor handled by NEB.

NOTE: The requesting service provider has a mandatory requirement of getting the correct consent from the end-user, before sending the end-user to another service provider with a MitID CT Exchange Code. The official description of the requirement is

> *When the user is performing a controlled transfer from service provider A to service provider B, it is the responsibility of service provider A to get a consent from the end user, regarding eID attributes which service provider A has requested on initial request, since these attributes will be available to service provider B.*

Requesting a Controlled Transfer Token Exchange Code

A Controlled Transfer Token Exchange Code is retrieved by calling the MitID Controlled Transfer Token Exchange Code endpoint defined here:

```
POST {Authority URL}/v1/mitid/controlledtransfer/tokenExchangeCode
 Authorization: Bearer [access token]
 Content-Type: application/json

 {
  ”targetBrokerId” : ”d2808b3b-d6dc-4258-9ba6-3c31fae76b5a”,
  ”targetServiceProviderId” : ”c6a346b2-1f7f-46de-826b-91639cd4a8eb”,
  ”transferTokenText” : ”transfer token text”
 }
```

Required scope	Description
mitid_api	It is required, that the client has requested the mitid_api scope to be allowed to invoke the MitID Controlled Transfer Token Exchange Code API.

Parameters	**Description**
targetBrokerId	MitID Broker ID of receiving MitID broker
targetServiceProviderId	MitID Service Provider ID of receiving service provider
transferTokenText	Type: String The calling service provider must specify a Transfer Token Text and hand this out to the receiving service provider.

Example of response

```
HTTP/1.1 200 OK
 Content-Type: application/json
 Cache-Control: no-cache, no-store

 {
  ”transferTokenExchangeCode”: ”38e195d6-14ad-4ed9-9f0b-d72a26c8ed95”
 }
```

Exchanging a Controlled Transfer Token Exchange Code to a MitID login

As a service provider integrating to NEB, it is possible to exchange a MitID CT Exchange Code received from another service provider for a MitID login.

The MitID CT Exchange Code is used by NEB in exchange of a MitID authentication token from the MitID APIs and as such enables NEB to automatically login an end-user in exchange for the MitID CT Exchange Code.

The MitID CT Exchange Code is passed as a parameter to the MitID identity provider in the NEB OIDC specification, alongside the received Transfer Token Text, and will be used to login the end-user based on the MitID session used to create the MitID CT Exchange Code in the first place.

Identity Provider parameters (mitid)	Description
transfer_token_exchange_code	Type: string The issued MitID Controlled Transfer Token Exchange Code Example: ”38e195d6-14ad-4ed9-9f0b-d72a26c8ed95”
transfer_token_text	Type: string (non-empty) A Controlled Transfer Token Text, which is up to the calling Service Provider to define and set. This text must be handed out to the receiving service provider together with the MitID Controlled Transfer Token Exchange Code.

Note: See general section on identity provider parameters for reference on how to set the parameters.

NemLog-In3

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

https://migrering.nemlog-in.dk/

NemLog-in is a 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.

Note that the full documentation and test-environments has not been made available yet. So this section is based on an expectation of the integration.

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**

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
ssn_store	Request permission to store the SSN requested with the ssn scope. If this scope is specified, the user will be given the option to allow the requesting service provider to store the end user SSN, ex the Danish CPR number. This will be shown to the end-user as a checkbox with the appropriate text. When requested the Userinfo endpoint will issue the following claim type: • ssn_store_consent The ssn_store_consent claim value can be one of • “allowed” “denied” (default)
nemlogin.priviledges	List of claims: • nemlogin.priv Only Danish public service providers can request this scope.

ID Token identity claims

Claim value	Possible values
identity_type	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.

Signature at NemLog-In3

The NemLog-In3 identity provider will provide the Danish general digital signature service, allowing both private MitID identities, NemLog-In3 professional identities and organizations to sign either an PAdES or XAdES.

When technical details and flow descriptions become available, we will update our documentation with this.

Nets eID Broker will support the signature flows and integration.

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
idp_values	**nemid**

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 sign_text. The type member is a string with one of the following values • text • pdf
private_to_business	Type: bool (default: false) If set to true, the end-user will be guided through the NemID Private to Business flow. This will guide the end-user to select one of the companies (CVR) for which the end-user is authorized to represent the company alone. The selected CVR-number will be set in the **nemid.auth_to_repr** claim.

Scope	**Description**
nemid	List of claims: • nemid.common_name • nemid.pid • nemid.rid • nemid.dn • nemid.ssn • nemid.email • nemid.cvr • nemid.auth_to_repr
transaction_claims	Maps the following claims to the Userinfo endpoint output. Transaction claims will only be available from the Userinfo endpoint with the specific access token issued from the end-user authentication mapping the relevant claims. These claims are not shared in a SSO with other services. • transaction_id • nemid.code_app_trans_ctx • nemid.sign_text • nemid.xmldsig
ssn	Social Security Number. List of claims: • dk.cpr

	Will trigger MitID Demo CPR user-interaction.
ssn_store	Request permission to store the SSN requested with the ssn scope. If this scope is specified, the user will be given the option to allow the requesting service provider to store the end user SSN, ex the Danish CPR number. This will be shown to the end-user as a checkbox with the appropriate text. When requested the Userinfo endpoint will issue the following claim type: • ssn_store_consent The ssn_store_consent claim value can be one of • "allowed" "denied" (default)

ID Token identity claims

Claim value	**Possible values**
Identity_type	• 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 **ssn_store** scope is requested, the end user will be able give consent allowing the service provider to store the ssn.

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 for NemID flows.

The specific parameters have not yet been defined, but it will allow the end-user to select between the available CVR-numbers available for which his or hers private NemID is "Allowed to Represent" the specified company.

This requires that the user enters his CPR number (or that the calling service is allowed for automatic CPR retrieval) after which the CVR-list can be retrieved from the Danish "Erhvervsstyrelsen" and the flow can be completed.

International identity providers by Nets E-Ident

This section describes the identity providers that are available via the NEB platform provided by Nets E-Ident.

All identity providers available from the Nets E-Ident setup, is also available using the NEB interface. NEB will help with the setup and take care of the integration to E-Ident and provide a flexible usage of the services provided by the E-Ident service.

This allows your integrations to utilize all the identity providers support by both NEB and E-Ident while utilizing all other features from the NEB platform.

Supported Nets E-Ident identity providers

- BankID Norway
- BankID on mobile Norway
- Buypass Norway
- BankID Sweden

All eIDs available from E-Ident, found at https://www.nets.eu/developer/e-ident/eids/Pages/default.aspx.

BankID Norway

Request parameter	Description
acr_values	Possible values • https://netseidbroker.dk/acr/idp/bankid_no/default
idp_values	**bankid_no**

Scope	Description
bankid_no	List of claims: • **bankid_no.birthdate** • **bankid_no.family_name** • **bankid_no.given_name** • **bankid_no.pid**
bankid_no_cert	List of claims: • bankid_no.certificate • bankid_no.certpolicyoid • bankid_no.cn • bankid_no.dn
ssn	Social Security Number. List of claims: • no.cpr
ssn_store	Request permission to store the SSN requested with the ssn scope. If this scope is specified, the user will be given the option to allow the requesting service provider to store the end user SSN, ex the Danish CPR number.

	This will be shown to the end-user as a checkbox with the appropriate text. When requested the Userinfo endpoint will issue the following claim type: • ssn_store_consent The ssn_store_consent claim value can be one of • "allowed" "denied" (default)

ID Token identity claims

Claim value	**Possible values**
identity_type	private
idp	**bankid_no**
acr	https://netseidbroker.dk/acr/idp/bankid_no/default
ial	https://netseidbroker.dk/acr/idp/bankid_no/default
amr	Not specified yet.

Handling of SSN

All companies that are allowed to handle social security numbers (SSN) can get this in return after a BankID identification.

Note: Remember to specify that you want to process SSN when ordering your BankID setup.

BankID on mobile Norway

Used by around 4 million Norwegians, BankID has become a household brand and a highly trusted digital identification service for Norwegian citizens.

Request parameter	**Description**
acr_values	https://netseidbroker.dk/acr/idp/**bankid_mobile_no**/default
idp_values	**bankid_mobile_no**

Scope	**Description**
bankid_mobile_no	List of claims: • **bankid_mobile_no.birthdate** • **bankid_mobile_no.family_name** • **bankid_mobile_no.given_name** • **bankid_mobile_no.pid**
bankid_mobile_no_cert	List of claims: • bankid_no.certificate • bankid_no.certpolicyoid • bankid_no.cn

	• bankid_no.dn
bankid_mobile_no_phone	List of claims: • **bankid_mobile_no.phone_number**
ssn	Social Security Number. List of claims: • no.cpr
ssn_store	Request permission to store the SSN requested with the ssn scope. If this scope is specified, the user will be given the option to allow the requesting service provider to store the end user SSN, ex the Danish CPR number. This will be shown to the end-user as a checkbox with the appropriate text. When requested the Userinfo endpoint will issue the following claim type: • ssn_store_consent The ssn_store_consent claim value can be one of • "allowed" "denied" (default)

ID Token identity claims

Claim value	**Possible values**
identity_type	private
idp	**bankid_mobile_no**
acr	https://netseidbroker.dk/acr/idp/bankid_mobile_no/default
ial	https://netseidbroker.dk/acr/idp/bankid_mobile_no/default
amr	Not specified yet.

Handling of SSN

All companies that are allowed to handle social security numbers (SSN) can get this in return after a BankID identification.

Note: Remember to specify that you want to process SSN when ordering your BankID setup.

Buypass Norway

The Norwegian eID Buypass issues Buypass ID in both mobile and on smart card.

Request parameter	**Description**
acr_values	https://netseidbroker.dk/acr/idp/buypass_no/default
idp_values	buypass_no

Scope	Description
buypass_no	List of claims: • buypass_no.birthdate • buypass_no.family_name • buypass_no.given_name • buypass_no.name • buypass_no.pid
buypass_no_cert	List of claims: • buypass_no.certificate • buypass_no.dn
ssn	Social Security Number. List of claims: • no.ssn
ssn_store	Request permission to store the SSN requested with the ssn scope. If this scope is specified, the user will be given the option to allow the requesting service provider to store the end user SSN, ex the Danish CPR number. This will be shown to the end-user as a checkbox with the appropriate text. When requested the Userinfo endpoint will issue the following claim type: • ssn_store_consent The ssn_store_consent claim value can be one of • "allowed" "denied" (default)

ID Token identity claims

Claim value	Possible values
identity_type	private
idp	buypass_no
acr	https://netseidbroker.dk/acr/idp/buypass_no/default
ial	https://netseidbroker.dk/acr/idp/buypass_no/default
amr	Not specified yet.

Handling of SSN

The social security number (SSN) of an end user can be returned if you are allowed to receive this.

BankID Sweden

Used by almost 8 million Swedes, BankID has become a household brand and a highly trusted digital identification and signing service for Swedish citizens. Almost 7 million has a mobile BankID and this eID was used in 96 % of logins and signings. It is also available as BankID on file and BankID on card.

Request parameter	Description
acr_values	The following is supported • https://netseidbroker.dk/acr/idp/bankid_se/default
idp_values	bankid_se

Scope	Description
bankid_se	List of claims: • bankid_se.birthdate • bankid_se.family_name • bankid_se.given_name • bankid_se.name • bankid_se.pid
bankid_se_cert	List of claims: • bankid_se.certificate • bankid_se.certpolicyoid • bankid_se.cn • bankid_se.dn
ssn	Social Security Number. List of claims: • se.ssn
ssn_store	Request permission to store the SSN requested with the ssn scope. If this scope is specified, the user will be given the option to allow the requesting service provider to store the end user SSN, ex the Danish CPR number. This will be shown to the end-user as a checkbox with the appropriate text. When requested the Userinfo endpoint will issue the following claim type: • ssn_store_consent The ssn_store_consent claim value can be one of • “allowed” “denied” (default)

ID Token identity claims

Claim value	Possible values
identity_type	private
idp	bankid_se
acr	https://netseidbroker.dk/acr/idp/bankid_se/default
ial	https://netseidbroker.dk/acr/idp/bankid_se/default
amr	One of

	• bankid_se.file • bankid_se.smartcard • bankid_se.mobile

Handling of SSN

A user's SSN is a part of the end user certificate and always available from a BankID login. The SSN is the same as the SERIALNUMBER part of the dn claim in the ID Token (OIDC) or the DN attribute in the assertion (SAML). An example of this:

CN=Olav Widen, OID.2.5.4.41=(180427 13.09) Olav Widen - BankID på fil, SERIALNUMBER=195310021935, GIVENNAME=Olav, SURNAME=Widen, O=Testbank A AB (publ), C=SE

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
16. [OIDC-SESSION]: ”OpenID Connect Session Management” - https://openid.net/specs/openid-connect-session-1_0.html
17. [OIDC-FRONT-CHANNEL]: ”OpenID Connect Front-Channel Logout” - https://openid.net/specs/openid-connect-frontchannel-1_0.html
18. [OIDC-BACK-CHANNEL]: ”OpenID Connect Back-Channel Logout” - https://openid.net/specs/openid-connect-backchannel-1_0.html
19. [TOKEN-EXCHANGE]: https://tools.ietf.org/html/rfc8693