SignaturGruppen

Technical reference

for service providers

Version 1.2 2021

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.

Nets eID Broker documentation documents

This document services as the primary source of technical documentation. The collected list of technical documents for NEB is listed here.

All documents can be found at https://broker.signaturgruppen.dk/

Title	**Description**
Nets eID Broker Technical Reference **[NEB-TECHREF]**	The primary source of technical documentation for the integration to Nets eID Broker (this document).
Nets eID Broker Frontend Guide **[NEB-FRONTEND]**	Examples and guides for the frontend integration to Nets eID Broker and OpenID Connect in general. Also covering mobile apps.
Nets eID Broker Sessions **[NEB-SESSIONS]**	Technical information about sessions in the NEB platform. Contains information about how NEB handles session and the various ways service providers can utilize and handle this in their integration.
Nets eID Broker OpenID Connect Intro **[NEB-INTRO]**	
Nets eID Broker Identity Providers **[NEB-IDP]**	Contains technical information about the available identity providers supported by NEB.

Changelog

Version 1.2

- Added sections “Implicit flow and CORS” and “Backend for Frontend Pattern”

Version 1.1

- Added description of general error “no_ctx”

Version 1.0

- Updated description of OCSP response and verification in transaction tokens.
- Removed incorrect description of nonce in transaction tokens. Nonce is set in ID token and can be correlated via transaction_id between ID token and transaction tokens if needed.
- Updated transaction token validation description of signing certificate

Version 0.9.10

- Moving general information about session and session handling to separate document “Nets eID Broker Sessions”. Removed several entries and sections from this document.
- Added section “Nets eID Broker documentation documents”.
- Removed information about Administration API and Privilege API, these will get their own documentation at a later stage, when ready for general integration.
- Moving general guides and introduction help for OpenID Connect to “Nets eID Broker OpenID Connect Intro”.
- Moving specific identity provider information to “Nets eID Broker Identity Providers”
- Added swagger endpoint description for Nets eID Broker API
- Added subsection “Choosing between ID token sub claim or identity provider specific identifiers”

Version 0.9.9.16

- Made it clearer, that the MitID parameter “reference_text” only is shown at the last authenticator step.
- Fixed the description of the MitID parameter “transaction_text”, which was invalid.
- NemID parameters “sign_text” and “sign_text_type” description updated.

Version 0.9.9.15

- Renamed incorrect example of NemID parameter for App transaction context to correct parameter name (code_app_trans_ctx)

Version 0.9.9.14

- Pruned old entries of “under development” and removed several entries about content and features not made available yet
- Added MitID identity provider section: “NemID PID claim for MitID flows”

Version 0.9.9.13

- Added documentation of HTML restrictions for transaction_text identity provider parameter.

Version 0.9.9.12

- Updated MitID identity provider parameter from psd2 to require_psd2 (incorrect in previous versions)
- Updated description of MitID transaction signing to be limitied to only signed requests.
- Updated MitID error codes
- Removed ssn_store scope and ssn_store_consent claim, as it is currently not supported in Nets eID Broker.

Introduction

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

The intended audiences are IT developers and IT architects.

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

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

Integrating to the Nets eID Broker overview

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

OpenID Connect

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

OpenID Connect is fully supported and thus enables the widest range of clients to benefit from the services offered including, but not limited to, legacy OAuth clients, mobile apps, CRM portals like Microsoft Dynamics and Single Page Applications (SPA) written in JavaScript (client application). Note for SPA type applications however, that we primarily support the Implicit flow for these types of clients to avoid the security implications of handling access tokens in-browser.

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.

See the “Nets eID Broker OpenID Connect Introduction” [NEB-INTRO] for a basic introduction to the OIDC integration to NEB.

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 is possible to control and configure branding, layout, and default settings for various parameters.

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.

The signed JWT can be signed with any supported algorithm, for instance HS256 using a symmetric client secret or RS256 using a private RSA key matching a registered client certificate secret for the client in question.

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** (no ui for authentication) • **login** (force authentication request) • **select_account** (enable idp selection) 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. If **none** is used, a request for automatic login based on the end-user session is requested. If the automatic login could not be honored, an error will be returned to the service. If **select_account** is used, the end-user will be allowed to select among the available identity providers for the given flow, even though the end-user has an active session with NEB. It does not trigger a forced reauthentication, so if the end-user selects the same identity provider for which he has an active session, he will be automatically reauthenticated.

Supported identity provider parameters. See the “Nets eID Broker Identity Providers” [NEB-IDP] 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 “Nets eID Broker Identity Providers” [NEB-IDP] 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.
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 / Step-up / Consent flows

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.

As a general mechanism, NEB will look for any changes in requested scopes that may require additional input from the end-user, such as consents or other actions. This will be automatically handled by NEB.

Some identity providers support specific flow such as step-up under certain conditions. NEB will normally reuse an existing session and ignore any identity provider specific parameters in the authentication request if a session already exists. To force processing of the identity provider specific parameters, utilize the prompt=login parameter, which will trigger a re-authentication. This re-authentication might then support step-up if a previous session exists or any other identity provider specific handling, which will be described for each identity provider.

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 or requested.

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 listed in this document or in the Identity Provider documentation [NEB-IDP].

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.

The end-user is redirected (GET) or posted (POST) back to the redirect URL requested for the authentication request honoring the **response_mode** parameter.

NEB will utilize one of the error codes defined in the specification most appropriate to the error in question and will set the error_description to a NEB- or identity provider specific error code defined here or in the "Nets eID Broker Identity Providers" [NEB-IDP],

Example response:

```
[client_redirect_uri]?error=
access_denied&error_description=mitid_user_aborted&state=xyz
```

Error codes

The error codes described here will be returned in the error_description parameter on error redirects or post backs to the redirect URI.

Note: Refer to the specific identity provider [NEB-IDP] for a list of specific error codes able to be returned from that identity provider.

General error codes

Error (error)	Error description (error_description)	Description
access_denied	internal_error	An unexpected internal error.
access_denied	user_aborted	User aborted
access_denied	no_ctx	No context or session found for the user. In some circumstances, we can redirect the user back to the service with this error code. This can be due to expired signed request, due to clicking back in browsers or similar problems.

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 (non-exhaustive):

- Maps to a list of user claims for the ID token and UserInfo endpoint
- Is mapped directly to the issued access token **scope** 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.

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.

Implicit flow and CORS

Nets eID Broker supports the implicit flow (id_token) but does not support setting up any CORS origin rules for integrating clients and has no allowed CORS rules specified as default.

NEB does not encourage pure frontend-based integrations due to security concerns and has a very limited and restricted functionality available for frontend clients.

Frontend-based clients can request and complete the implicit flow but will have to pin the signing key certificate for verifying the ID token and will not be allowed accessing access tokens nor the Userinfo endpoint from the browser.

See the “Backend for Frontend Pattern” for reference for a recommended approach to securing your frontend and integration to NEB.

Backend for Frontend Pattern

Backend for Frontend (BFF) is a pattern utilizing an API-based backend for your frontend which enables your frontend applications to be simplified, more secure and enables the full set of available features of your backend like back-channel logout, proper session handling, API proxying etc.

See Duendes blog about this pattern here: https://blog.duendesoftware.com/posts/20210326_bff/

For .Net core backends, see here for references: https://docs.duendesoftware.com/identityserver/v5/bff/

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.

Preproduction environment – https://pp.netseidbroker.dk/op

Variable	Values
Authority URL	https://pp.netseidbroker.dk/op
Discovery endpoint	https://pp.netseidbroker.dk/op/.well-known/openid-configuration
Nets eID Broker MitID Broker ID	f81b4f9a-2ca2-49ec-ba52-654de7edfcdc

Token signing certificate

Subject	CN = Nets eID Broker Token Signing 1 PP Env C = DK

Subject Thumbprint (KID)	**048058BB59F4D3007045896FD488CE81F4EB4923**
CA Subject	CN = Nets eID Broker Token Signing Root PP Env C = DK
CA Thumbprint	1beb2d3df149237427ae40abe524882a7ebb2ddb

SSL certificate

Subject	CN = pp.netseidbroker.dk
CA Subject	CN = R3 O = Let's Encrypt C = US
CA Thumbprint	a053375bfe84e8b748782c7cee15827a6af5a405

Transaction token signing certificate

Subject	CN = SIGNATURGRUPPEN A/S - NEB Transact PP
	SERIALNUMBER = CVR:29915938-UID:59911227
	O = SIGNATURGRUPPEN A/S // CVR:29915938
	C = DK

Subject Thumbprint (KID)	20595A4BE9F566771792BC3DBC7DF78FF9C36575
CA Subject	CN = TRUST2408 Systemtest XXXIV CA O = TRUST2408 C = DK
CA Thumbprint	eeaf09230cd54e31a22872bd83cd189095921ad7

Production environment – https://netseidbroker.dk/op

Variable	**Values**
Authority URL	https://netseidbroker.dk/op
Discovery endpoint	https://netseidbroker.dk/op/.well-known/openid-configuration
Nets eID Broker MitID Broker ID	a9df260d-42c6-4e4c-85a5-681423673a78

Token signing certificate

Subject	CN = Nets eID Broker Token Signing 1 Prod Env C = DK
Subject Thumbprint (KID)	353E2FE9191CDEC22C8B52D2B7A82A2DAA50642E
CA Subject	CN = Nets eID Broker Token Signing Root Prod Env C = DK
CA Thumbprint	fa516c6bb2d07103a54fe4cd6ded4aed30b360f7

SSL certificate

Subject	CN = netseidbroker.dk
CA Subject	CN = R3 O = Let's Encrypt C = US
CA Thumbprint	a053375bfe84e8b748782c7cee15827a6af5a405

Transaction token signing certificate

Subject	CN = SIGNATURGRUPPEN A/S - eID Broker Signing
	SERIALNUMBER = CVR:29915938-UID:14521394
	O = SIGNATURGRUPPEN A/S // CVR:29915938
	C = DK

Subject Thumbprint (KID)	8CB7F2CBABA3A57979DF96BC81DC0EAF44F30F9B
CA Subject	CN = TRUST2408 OCES CA IV O = TRUST2408 C = DK
CA Thumbprint	5084ef33f0d4a39776281ccfdf0a9b06eea7fb9a

Nets eID Broker API

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

Swagger endpoint

Swagger endpoint URL	Description
[Authority URL]/swagger/index.html	The swagger description of the available Nets eID Broker API

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 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.

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.identity_name"                : "Hans Hansen",
 "session_status"                     : "active",
 "session_identifier"                 : "19712D37-0C76-4AAA-B049-BACD585F484F"
}
```


Logout and sessions

Refer to [NEB-SESSIONS] for details on how to log out the end-user and various ways to utilize the standard sessions of NEB.

Refer to the Broker API swagger endpoint defined above, for technical details on the API endpoints.

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 also/only get the ID token 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.
neb_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 **neb_sid** values are used to identify distinct sessions at NEB. The **neb_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. This identifier will be unique for each receiving organization for each associated identity. This means that all services under the same organization (defined in NEB admin-UI) will receive the same sub claim for the same identity (i.e. MitID identity). But services under different organizations will receive a different sub claim. More details are described in the Organization specific subject claims section.
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.
idp	Identity provider who issued the underlying identity. Refer to the specific identity providers for a list of possible **idp** values.
idp_environment	Optional available. See the different identity provider sections for specific details. If set, this specifies the environment for the identity provider specified in the idp claim.
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.
session_expiry	Absolute end-user session expiry represented as seconds elapsed since 00:00:00 UTC on 1 January 1970 (Unix timestamp) This value represents the time when the end-user session expires. Services may not use the issued authentication ticket from NEB to create new sessions after this time.

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,
```

```
"loa"                    : "https://data.gov.dk/concept/core/nsis/Low",
"amr"                                : ["mitid.password"],
"identity_type"          : "private",
"idp"                    : "mitid",
"iat"                    : 1311290550,
"exp"                    : 1311291550,
"...."                   : ....,
}
```

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.

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 (note that this response is currently under development).

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.
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.

Organization specific subject claims

The subject claim, **sub**, issued in all ID tokens are the primary identity identifier and is always a unique organization specific GUID representing the identity for the receiving organization.

All services under the same organization, defined in the NEB administrative interface, will receive the same sub claim from the same identity.

This enables all service within an organization to map the same sub claim to the same identity.

If the same identity logs into a service under a different organization NEB will issue a different sub claim for this service/organization.

If a more global identifier is needed, the different identity providers will typically allow for identity provider specific identifiers to be requested such as the globally scoped MitID UUID – which will be available via the Userinfo endpoint.

Choosing between ID token sub claim or identity provider specific identifiers

It is a common problem for an integrating service to decide how to map the incoming user from the incoming claims to a local user context.

Taking MitID as an example (replace with any specific identity provider), the choice is between registering a new user using the sub claim from the ID token or using the mitid.uuid claim, available from the Userinfo endpint.

Using the sub claim allows for minimizing the sensitivity of the received identifiers and information about the end-user, whereas the mitid.uuid claim is a global identifier providing a perhaps more useful identifier.

On a privacy note, it is always recommended to use the ID token sub, as this has the lowest impact on traceability of the end-user. On the other hand, mitid.uuid is perhaps required if a global identifier of the end-user is needed.

If only the ID token sub claim is needed, then a authentication flow does not even have to invoke the Userinfo endpoint after completion, thus minimizing both traffic and exchanged userinfo.

The recommendation is to consider whether the ID token sub can be used as the primary identifier for the end-users and only if this consideration determines that it is not sufficient, a move to the available global identifiers of the end-users should be considered.

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 can 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 **amr**, **idp** and **identity_type** are as expected.
- The client MUST validate that the expected restrictions for identity provider specific claims are as expected. Example could be MitID claims: loa, ial and aal.

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 the certificate specified as the “Transaction token signing certificate” 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 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 expected restrictions for **acr, ial**, **amr**, **idp** and **identity_type** are as expected.
- Optionally, validate the OCSP response of the signing certificate, set as transaction_token_ocsp_resp in the token response (currently not implemented, will be released soon).

- Optionally, validate the NONCE parameter by correlating the transaction_id found in the transaction token with the ID token, as the NONCE parameter is only set in the ID token.

Verification of UserInfo endpoint response

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

Identity Providers

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, "code_app_trans_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.

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
20. [ERROR-RESPONSES]: https://tools.ietf.org/html/rfc6749#section-4.1.2.1
21. [CLIENT-AUTHENTICATION]: https://openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication