Table of Contents

Identity Hub - Release Notes

The following versions of Connective Identity Hub are currently supported:

- Identity Hub 1.0
- Identity Hub 1.0.1

Identity Hub 1.0 - Release Notes

Release date: 2020-06-19

1.1 New features

The Connective Identity Hub is Connective's new identification service based on the OpenID Connect protocol.

With Connective Identity Hub 1.0, customers can use different identification providers such as eID and itsme to identity their users. In future versions, more identification providers will be added.

1.2 Handled issues

N/A.

1.3 Known issues

N/A.

Identity Hub 1.0.1 - Release Notes

Release date: 2020-09-25

1.1 New features

Identity Hub 1.0.1 is a hotfix version and doesn't contain new features.

1.2 Handled issues

JIRA CODE	ISSUE CODE	DESCRIPTION
CIH-14	/	Date and time are now added to the logs.
CIH-35	/	Canceling the readout in IDS sometimes resulted in an endless loop.
CIH-38	/	Incoming json fields from Itsme are now correctly parsed.

1.3 Known issues

N/A.

Identity Hub 1.0 - Integration Guide

Introduction

This section describes the technical integration with the Connective Identity Hub.

The API for the Identity Hub is based on the OpenID Connect protocol, specifically the Authorization Code Flow. The Connective implementation is a basic implementation of this protocol. This document will describe the protocol as implemented for the Connective Identity Hub.

The OpenID Connect protocol goes as follows:

- The customer application redirects the end user's browser to the Identity Hub's Authorization Endpoint, indicating what user information will be requested.
- The Identity Hub redirects to the Authorization Endpoint of the requested Identity Provider.
- The user identifies himself by a means supported by the Identity Provider.
- The Identity Hub redirects the user's browser to customer application with a "code".
- The customer application's backend calls the Token Endpoint and authorizes itself, in exchange the customer application receives an "access token" (to get access to the user's data) and an "id token" which mainly serves as an identifier for the user.
- The customer application's backend calls the UserInfo endpoint to retrieve the requested user information formatted as claims.

The Identity Hub also implements the Jwks Endpoint which lets the customer application retrieve all public keys necessary to verify JWT tokens which were issued by the service.

The OpenID Connect Discovery endpoint can be called to retrieve information of where all the other endpoints are located and what modes are supported.

Table of contents

1. Getting started

Some customer information needs to be collected before the *Connective Identity Hub* can be used:

Each client application from the customer needs to define a **name**.

Another necessary bit of information is the allowed **redirect URLs**. The Connective Identity Hub is based on the OpenID Connect protocol. OpenID Connect works by redirecting the user between the customer application and the Identification Service. While the user information cannot be retrieved without a password, an attacker may still try to initiate a readout flow and redirect the end user to a phishing website. Each client should therefore specify **up-front** which redirect URLs will be allowed. An error will be raised if the redirect URL in a given parameter does not match the ones registered in the configuration. A redirect URL should use the HTTP/S scheme, cannot contain any query parameters and must always be passed to the *Connective Identity Hub* in the exact same way it was registered.

The customer should also indicate which Identity Providers the client should be able to use. It is possible that additional information will be needed when making use of a specific Identity Provider.

Connective can then proceed to initialize the configuration. Once that is done, each client will receive a unique OpenId Connect **client_id** associated with the client's redirect URL and a **unique client_secret**.

The *Connective Identity Hub* allows two **authentication methods** to be used for communication with the customer's client application's backend:

- Basic authentication: encodes the client_secret and transmits it through the Authorization HTTP header
- *POST* authentication: sends, amongst others, the client_secret in a form-url-encoded POST body

Which method you should use depends largely on the OpenId Connect library used by the customer application. Either method will have to communicate over an HTTP/S connection, so the password is never sent "in the clear". Basic authentication is preferred because it allows the HTTP server to quickly filter bad requests.

2. Authorization endpoint

Request

The authorization endpoint is the entry point to start an identification process. The customer's client application redirects the user's browser to this entry point:

```
https://<servername>/connect/authorize
```

This URL can be retrieved from the Discovery Endpoint, using the key *authorization_endpoint*.

Request query parameters:

PARAMETER	DESCRIPTION	USE
response_type	This defines the processing flow to be used when forming the response. Because *Connective Identity Hub* uses the Authorization Code Flow, this value **must** be **code**.	Required
client_id	Identifier for the client customer application, supplied by Connective.	Required
acr_values	Space-separated string that specifies the acr values that the Authorization Server is being requested to use for processing this Authentication Request, with the values appearing in order of preference. This parameter will be used to decide on the actual Identity Provider that will be called. For this reason, the value **must** contain ***idp:<name_of_identity_provider>.***	Required
redirect_uri	URL where the *Connective Identity Hub* will redirect the user's browser to with the intent to deliver a ***code*** to the customer's client application when successful, or to inform it about errors. Must match the client's configuration as seen in section Getting Started. **Note:** when using https://localhost as redirect_uri, the response_mode=form_post must be used	Required
scope	The scope parameter allows the application to express the desired scope of the access request. It **must** contain the value ***openid***. You may also specify additional scopes, separated by spaces, to request more information about the user. Please read the documentation of the specific Identity Provider to get the full list of supported scopes.	Required
state	An opaque value used in the Authentication Request, which will be returned unchanged in the Authorization Code. This parameter should be used for preventing cross-site request forgery (XRSF).	Recommended
nonce	A string value used to associate a session with an Id Token, and to mitigate replay attacks. The value is passed through unmodified from the Authentication Request to the Id Token. Sufficient entropy must be present in the *nonce* values used to prevent attackers from guessing values.	Recommended
claims	This parameter is used to request specific claims. The value is a JSON object listing the requested claims. Please read the documentation of the specific Identity Provider to get the full list of supported claims.	Optional
ui_locales	Language code to indicate in which language to present the UI. Please read the documentation of the specific Identity Provider to get the full list of supported locales.	Optional

PARAMETER	DESCRIPTION	USE
code_challenge	A challenge derived from the code verifier that is sent in the authorization request, to be verified against later. A code verifier is a cryptographically random string that is used to correlate the authorization request to the token request. Use this parameter to initiate a PKCE (Proof Key for Code Exchange) flow.	Optional
code_challenge_method	The method that was used to derive code challenge. It **must** contain the value **S256** *if a code_challenge*, and thus the PKCE (Proof Key for Code Exchange) flow, is used. The challenge is calculated as follows: *code_challenge* = BASE64URL-ENCODE(SHA256(ASCII(*code_verifier*)))	Conditional

Response

Success

If the user is successfully authenticated and authorizes access to the requested data, *Connective Identity Hub* will return an Authorization Code to your server component. This is achieved by returning an Authentication Response, which is a HTTP 302 redirect request to the ***redirect_uri*** specified previously in the Authentication Request.

The response will contain following query parameters:

PARAMETER	DESCRIPTION	USE
code	The ***code*** parameter holds the Authorization Code, which is a string value. This value should be provided to the Token endpoint as described in the Token Endpoint section.	Required
state	The ***state*** parameter will be returned *if* a value was provided in the Authentication Request. The returned value should match the one supplied in the Authentication Request.	Conditional

Error

If the request fails due to a missing, invalid, or mismatching redirection URI, or if the client identifier is missing or invalid, the Identity Provider should inform the user of the error and must not automatically redirect him to the invalid redirection URI.

If the user denies the Authentication Request or if the request fails for reasons other than a missing or invalid redirection URI, the *Connective Identity Hub* will return an error response to your client application. As for a successful response this is achieved by returning a HTTP 302 redirect request to the ***redirect_uri*** specified in the Authentication Request.

The error response query parameters are the following:

PARAMETER	DESCRIPTION	USE
error	The error type	Required
error_description	A description of the error, indicating the problem in more detail.	Optional
state	The ***state*** parameter will be returned *if* a value was provided in the Authentication Request.	Conditional

3. Token Endpoint

Request

When calling the Token endpoint, the customer's client application needs to authenticated by using its unique **client_id** and **client_secret**. As mentioned in section Getting Started, there are 2 methods of providing these credentials. In both cases the HTTP Method is POST. The content type should be set to **application/x-www-form-urlencoded**.

The Token endpoint can be found at the following URL:

```
https://<servername>/connect/token
```

This URL can be retrieved from the Discovery Endpoint, using the key *token_endpoint*.

The *Connective Identity Hub* supports the following parameters in the POST body:

PARAMETER	DESCRIPTION	USE
grant_type	This **must** be set to ***authorization_code***.	Required
code	The Authorization Code received in response to the Authentication Request.	Required
redirect_uri	The redirection URI supplied in the original Authentication Request. This is the URL to which you want the user to be redirected after the authorization is complete.	Required
code_verifier	A code verifier is a cryptographically random string that is used to correlate the authorization request to the token request. Use this parameter when a *code_challenge* was used in the authentication request (PKCE flow).	Conditional

Response

Success

If the Token Request has been successfully validated, we will return an HTTP 200 OK response including id and access tokens. Content type will be ***application/json***.

The response body will include:

PARAMETER	DESCRIPTION	USE
access_token	The access token which may be used to access the Userinfo Endpoint.	Required
expires_in	The number of seconds the **access_token** will remain valid.	Required
token_type	Set to ***Bearer***	Required
id_token	The id token is a JSON Web Token (JWT) that contains user profile information represented in the form of claims.	Required

With the following values returned in the ***id_token***:

PARAMETER	DESCRIPTION	USE
iss	Identifier of the issuer of the Id Token.	Required

PARAMETER	DESCRIPTION	USE
sub	An identifier for the user. Use *sub* in the customer's client application as the unique identifier key for the user.	Required
aud	Audience of the Id Token. This will contain the *client_id*. This is the client identifier you received when registering your customer client application in the *Connective Identity Hub*.	Required
exp	Expiration time on or after which the Id Token **must not** be accepted for processing.	Required
nbf	The time before which the Id Token **must not** be accepted for processing.	Required
iat	The time the Id Token was issued, represented in Unix time (integer seconds).	Required
auth_time	Time when the End-User authentication occurred, represented in Unix time (integer seconds).	Required
amr	This value will be set to ***external***.	Required
idp	This value will be set to the name of the Identity Provider that was used to perform the authentication request.	Required
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. *If* present in the Id token, clients **must** verify that the nonce claim value is equal to the value of the nonce parameter sent in the authentication request.	Conditional
at_hash	Access Token hash value.	Required
s_hash	State hash value. *If* a *state* parameter was sent in the authentication request.	Conditional

Error

If the Token Request is invalid or unauthorized an HTTP 400 response will be returned. Content type will be ***application/json***.

The error response body contains:

PARAMETER	DESCRIPTION	USE
error	The error type.	Required
error_description	A description of the error, indicating the problem in more detail.	Optional

4. Userinfo Endpoint

The customer's client application's backend calls the UserInfo Endpoint to retrieve the user info in claims form.

To do so, the customer's client application needs to provide the **access_token** as received from the Token Endpoint in the HTTP Authorization header. The access token is only valid for a limited amount of time so this call must be done before the token is expired.

The UserInfo Endpoint can be found at the following URL:

```
https://<servername>/connect/userinfo
```

This URL can be retrieved from the Discovery Endpoint, using the key *userinfo_endpoint*.

Success

The Userinfo Endpoint will return an HTTP 200 response and the user claims in a plain JSON format. Content type will be ***application/json***.

The actual contents of the response depend on the scopes given to the Authorization Endpoint and the selected Identity Provider.

Error

If the Token Request is invalid or unauthorized a response will be returned. Content type will be ***application/json***.

The error response status code can be:

STATUS	DESCRIPTION
400	In case of invalid request
401	In case of expired / invalid token
403	In case of insufficient scope

The error response body will contain:

PARAMETER	DESCRIPTION	USE
error	The error type.	Required
error_description	A description of the error, indicating the problem in more detail.	Optional

5. Discovery Endpoint

The *Connective Identity Hub* implements the OpenId Connect Discovery Endpoint to find out all the URIs of the endpoints that are available.

The Discovery Endpoint can be found at the following URL:

```
https://<servername>/.well-known/openid-configuration
```

The HTTP method is GET, no authentication is necessary.

Content type of the response will be ***application/json***.

6. JWKS Endpoint

In order to validate the signatures of JWTs returned by the *Connective Identity Hub*, the JWKS Endpoint can be called to retrieve the public keys. The identifier of the particular key, which was used in a signature, will be present in a header of the JWT.

The JWKS Endpoint can be found at the following URL:

```
https://<servername>/.well-known/openid-configuration/jwks
```

The HTTP method is GET, no authentication is necessary.

Content type of the response will be ***application/json***.

7. Supported Identity Providers

Depending on the configuration of the client application, one of the following **Identity Providers** can be active.

Reach out to service@connective.eu if an IDP has not been activated for your client application

Belgian eID

ACR_VALUES	IDP:IDS
scopes	https://documentation.connective.eu/en-us/IdentificationService/SupportedScopesandReturnedClaims.html
claims	N/A
ui_locales	nl-be, fr-fr, en-us, de-de

itsme® Identification Service

ACR_VALUES	IDP:ITSME
scopes	https://belgianmobileid.github.io/slate/v2/identification.html#Data
claims	https://belgianmobileid.github.io/slate/v2/identification.html#Data
ui_locales	nl, fr, en, de