OpenID Connect explained

connect2id

CONNECTING APPS TO IDENTITIES

What is

OpenID Connect?

OpenID Connect is a new internet standard for

Single Sign-On (SSO)

Identity Provision (IdP)

OpenID Connect supports

web clients

mobile / native clients

OpenID Connect is good for

consumer apps

social apps

enterprise apps

mobile apps

OpenID Connect is backed by

- Microsoft
- Google
- Oracle
- Ping
- Salesforce
- … us and many others

OpenID Connect distilled

1. Need to authenticate a user?

2. Send user to their OpenID provider
   (via browser / HTTP 302 redirect)

3. Get identity token back

The key OpenID Connect object

ID Token

asserts the user's identity (user ID)

Client apps receive an ID token from the OpenID Provider

ID token

CARNET INTERNACIONAL DE ESTUDIANTE
INTERNATIONAL STUDENT IDENTITY CARD
isic
Studies at | Étudiant à | Est. de Enseñanza
University of Economics
Name | Nom | Nombre
B. Smart
Born | Né(e) le | Nacido/a el
09/03/1993
Validity | Validité | Validez
UNESCO
S 123 456 789 012 X
CARTE D'ÉTUDIANT INTERNATIONALE

The ID token resembles the concept of an **identity card**, in a standard digital format that client apps can validate.

- Asserts the user's identity.
- Has an issuing authority (the IdP).
- May specify how (strength, factors) and when the user was authenticated.
- Is generated for a particular audience (client).
- Has an issue and an expiration date.
- May contain details such as the user's name, email address and other profile information.
- Is digitally signed, so the intended recipients can verify it.
- May optionally be encrypted for confidentiality.

ID token internals

```
{
 "iss"    : "https://c2id.com",
 "sub"    : "alice",
 "aud"    : "s6BhdRkqt3",
 "nonce"  : "n-0S6_WzA2Mj",
 "exp"    : 1311281970,
 "iat"    : 1311280970,
 "acr"    : "http://loa.c2id.com/high",
 "amr"    : [ "mfa", "pwd", "otp" ]
}
```

- Encoded as a JSON Web Token (JWT).
- The claims about the authenticated end-user (subject) are packaged in a simple JSON object.
- Digitally signed with the OpenID Provider's RSA or EC key.
- Is URL-safe.

Encoded ID token

```
eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzcyI6ICJodHRw
Oi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5NzYxMDAxIiw
KICJhdWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAibi0wUzZfV3pBMk1qIi
wKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEzMTEyODA5NzAKfQ.ggW8hZ
1EuVLuxNuuIJKX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU96Sr_P6qJp6IcmD3HP9
9Obi1PRs-cwh3LO-p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJNqeGpegccMg
4vfKjkM8FcGvnzZUN4_KSP0aAp1tOJ1zZwgjxqGByKHiOtX7TpdQyHE5lcMiKPX
fEIQILVq0pc_E2DzL7emopWoaoZTF_m0_N0YzFC6g6EJbOEoRoSK5hoDalrcvRY
LSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfubEEBLuVVk4XUVrWOLrLl0n
x7RkKU8NXNHq-rvKMzqg
```

[Header] . [Claims] . [Digital Signature]

Cool ID token uses

- Simple stateless session management for JavaScript and single-page applications.
- Universal passport for all your users and applications, regardless of where they came from – social networks, partner businesses and organisations, local accounts.
- May be passed to 3rd parties to assert the user's identity.
- May be exchanged for an access token at the token endpoint of an OAuth 2.0 authorisation server. See draft-ietf-oauth-token-exchange-05.

How to obtain an ID token?

Via the OAuth 2.0

protocol flows

Choose an OAuth 2.0 flow to suit your app

- **Authorisation code flow**
  - for typical web and mobile apps
  - the client is typically authenticated
  - tokens retrieved via back channel
- **Implicit flow**
  - for JavaScript applications that run in the browser
  - the client is **not** authenticated
  - tokens returned via front-channel, revealed to browser
- **Hybrid flow**
  - allows app front-end and back-end to receive tokens independently
  - rarely used

http://openid.net/specs/openid-connect-core-1_0.html#Authentication

The OpenID auth request (code flow)

Send the user to the OpenID provider with an authentication request:

```
https://openid.provider.com/authorize?
    response_type=code
    &scope=openid
    &client_id=s6BhdRkqt3
    &state=af0ifjsldkj
    &redirect_uri=https%3A%2 %2Fclient.example.org%2Fcb
```

The OpenID auth response (code flow)

If the user is successfully authenticatted the OpenID provider will redirect the browser back to the client app with an authorisation code:

```
https://client.example.org/cb?
    code=SplxlOBeZQQYbYS6WxSbIA
    &state=af0ifjsldkj
```

The OpenID auth response (code flow)

If the authentication request cannot be fulfilled for some reason the OpenID provider may return an error code:

```
https://client.example.org/cb?
    error=access_denied
    &state=af0ifjsldkj
```

Exchange code for ID token (code flow)

Makes a back channel request to exchange the code for an ID token. Note that the client authenticates itself to the server here!

```
POST /token HTTP/1.1
Host: openid.provider.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

grant_type=authorization_code
 &code=SplxlOBeZQQYbYS6WxSbIA
 &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

Exchange code for ID token (code flow)

Finally, we have our ID token! But what's this access token?

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

{
   "access_token": "SlAV32hkKG",
   "token_type": "Bearer",
   "refresh_token": "8xLOxBtZp8",
   "expires_in": 3600,
   "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazc..."
}
```

UserInfo

```
{
   "sub"            : "alice",
   "name"           : "Alice Adams",
   "given_name"     : "Alice",
   "family_name"    : "Adams",
   "email"          : "alice@wonderland.net",
   "email_verified" : true,
   "phone_number"   : "+359 (99) 100200305",
   "profile"        : "https://c2id.com/users/alice",
   "ldap_groups"    : [ "audit", "admin" ]
 }
```

OpenID Connect defines an extensible JSON schema for releasing consented user details to client applications

Requesting UserInfo with the OpenID auth request

Send user to OpenID provider with auth request:

```
https://openid.provider.com/authorize?
    response_type=code
    &scope=openid%20profile%20email
    &client_id=s6BhdRkqt3
    &state=af0ifjsldkj
    &redirect_uri=https%3A%2 %2Fclient.example.org%2Fcb
```

Access token

Resembles the concept of a physical token or ticket. Permits the bearer access to a specific resource or service. Has typically an expiration associated with it.

- OAuth 2.0 access tokens are employed in OpenID Connect to allow the client application to retrieve consented user details from a UserInfo endpoint.
- The server may extend the access token scope to allow the client access to other protected resources and web APIs.
- The client treats the access token as a simple opaque string to be passed with the HTTP request to the protected resource.

UserInfo request with access token

Put the obtained bearer token in the authorization header of your outgoing HTTPS request:

```
GET /userinfo HTTP/1.1
Host: server.example.com
Authorization: Bearer SlAV32hkKG
```

UserInfo response

Sample response from the UserInfo endpoint, with the consented details (claims / assertions) about the user:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "sub"            : "alice",
  "name"           : "Alice Adams",
  "email"          : "alice@wonderland.net",
  "email_verified" : true,
  "phone_number"   : "+359 (99) 100200305",
  "profile"        : "https://c2id.com/users/alice",
  "ldap_groups"    : [ "audit", "admin" ]
 }
```

The two OpenID Connect tokens summed up

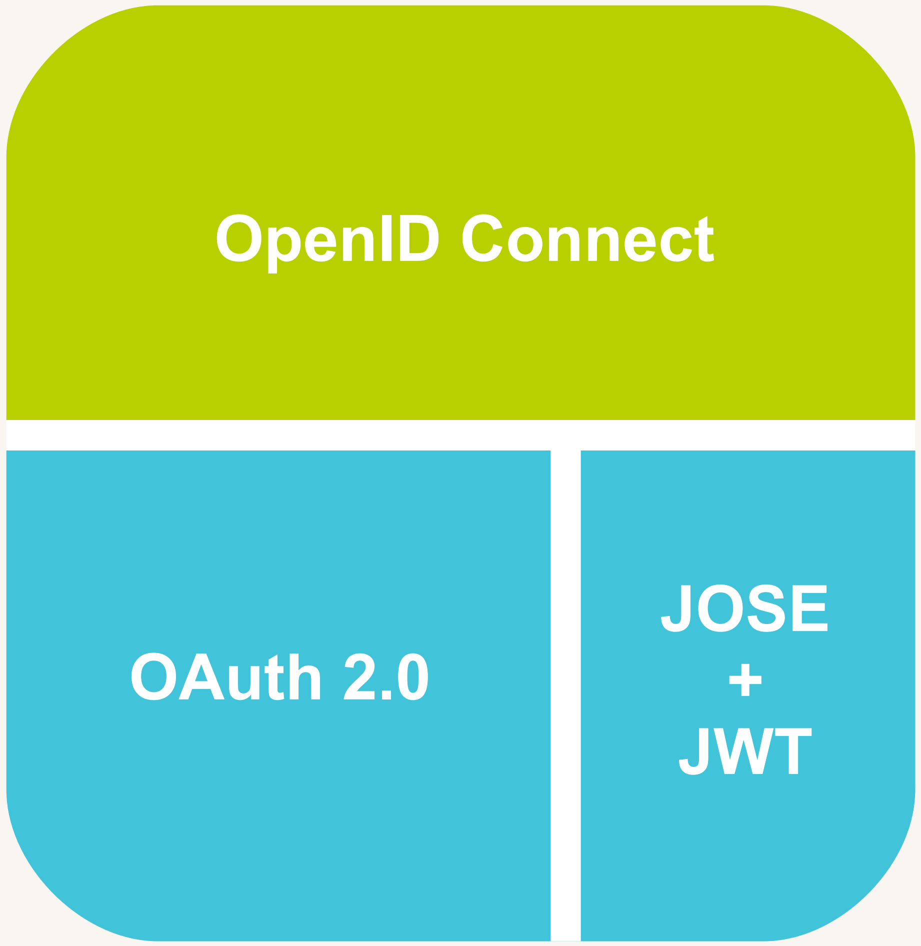
ID Token

asserts the user's identity (user ID)

Access Token

optional, to retrieve consented UserInfo

OpenID Connect rides on top of OAuth 2.0



- User identity is asserted by means of JSON Web Tokens (JWT)
- Clients use standard OAuth 2.0 flows to obtain ID tokens
- Guiding mantra: Simple clients, complexity absorbed by the server
- Any method for authenticating users – LDAP, tokens, biometrics, etc.
- JSON schema for UserInfo
- Supports optional OpenID provider discovery, dynamic client registration and session management.
- Extensible to suit many use cases.
- Federation is possible.

OpenID Connect provider endpoints

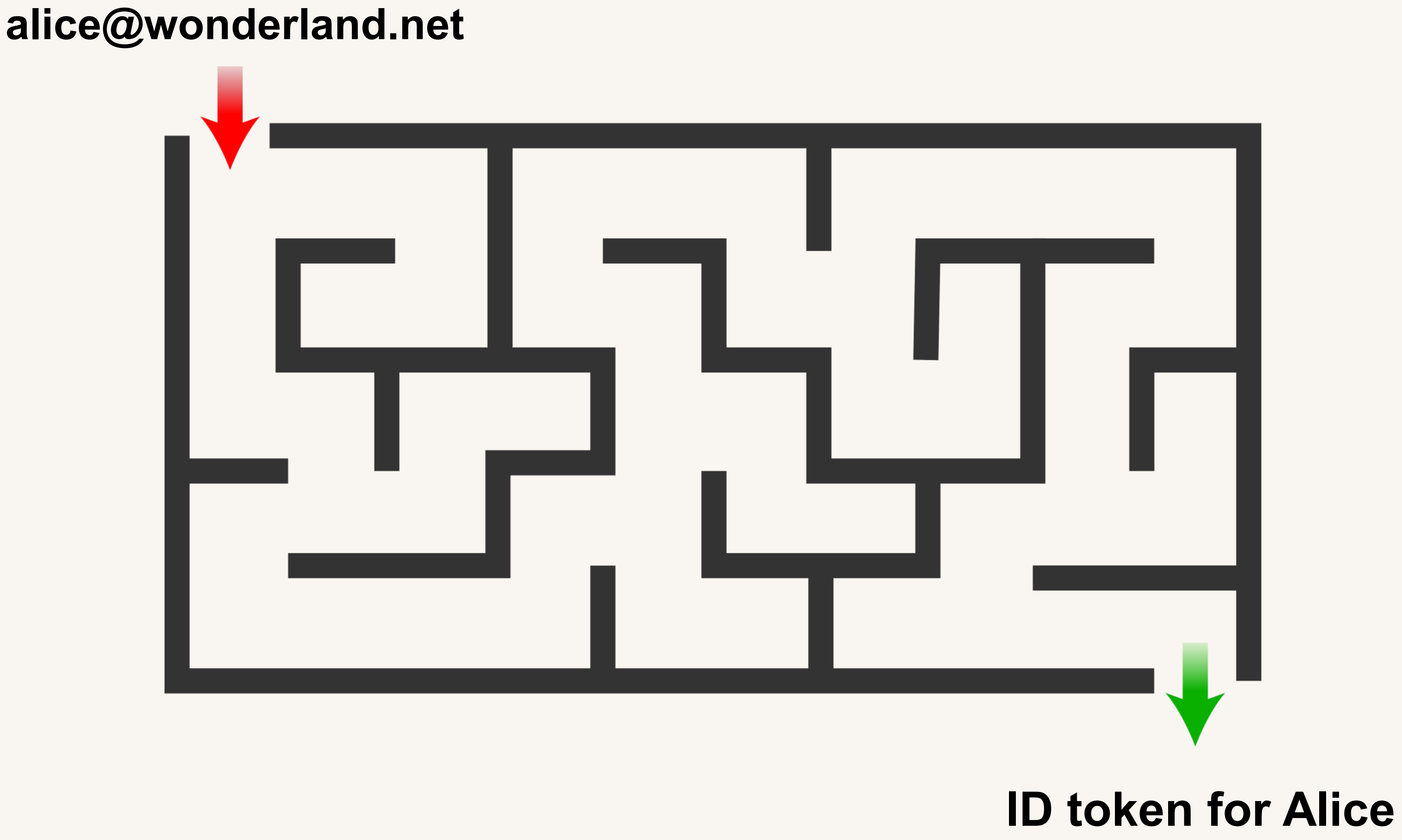
HTTP Endpoints

- Core provider endpoints:
  - Authorisation endpoint
  - Token endpoint
  - UserInfo endpoint
- Optional provider endpoints:
  - WebFinger endpoint
  - Provider metadata URI
  - Provider JWK set URI
  - Client registration endpoint
  - Session management endpoint

Optional endpoints

- WebFinger : Enables dynamic discovery of the OpenID Connect provider for a user based on their email address.
- Provider configuration URI : Well-known URL of a JSON document advertising the endpoints and capabilities of the OpenID provider. Helps the client apps to auto-configure their OpenID Connect requests.
- Provider JWK set URI : JSON document containing the OpenID provider's public (typically RSA) keys in JSON Web Key (JWK) format. These keys are used to sign the issued ID tokens and other artefacts.
- Client registration : Enables client apps to register dynamically, then update their details or unregister. Registration may be open (public).
- Session management : Enables client apps to check if a logged in user has an active session with the OpenID provider. Also to signal logout.

The future: dynamic discovery + client registration



The specifications

- OpenID Connect: http://openid.net/connect
- OAuth 2.0 (RFC 6749): http://tools.ietf.org/html/rfc6749
- OAuth 2.0 Bearer token (RFC 6750): http://tools.ietf.org/html/rfc6750
- JSON Web Token: http://tools.ietf.org/html/rfc7519
- JSON Web Signature: http://tools.ietf.org/html/rfc7515
- JSON Web Encryption: http://tools.ietf.org/html/rfc7516
- JSON Web Key: http://tools.ietf.org/html/rfc7517

Thank You!

Q + A

Get these slides from

http://connect2id.com/assets/oidc-explained.pdf