Securing Our Web Application

Kevin Dockx
ARCHITECT

@KevinDockx https://www.kevindockx.com

Coming Up

The Hybrid Flow

Logging in, Logging out

Getting Identity Claims

```
https://idphostaddress/connect/authorize?
client_id=imagegalleryclient
&redirect_uri=https://clientapphostaddress/signin-oidc
&scope=openid profile
&response_type=code id_token
&response_mode=form_post
&nonce=63626...n2eNMxA0
```

The Hybrid Flow

Authentication request to the authorization endpoint

```
https://idphostaddress/connect/authorize?
client_id=imagegalleryclient
&redirect_uri=https://clientapphostaddress/signin-oidc
&scope=openid profile
&response_type=code id_token
&response_mode=form_post
&nonce=63626...n2eNMxA0
```

The Hybrid Flow

Authorization endpoint at IDP level

```
https://idphostaddress/connect/authorize?
client_id=imagegalleryclient
&redirect_uri=https://clientapphostaddress/signin-oidc
&scope=openid profile
&response_type=code id_token
&response_mode=form_post
&nonce=63626...n2eNMxA0
```

The Hybrid Flow

Identifier of the client

```
https://idphostaddress/connect/authorize?
client_id=imagegalleryclient
&redirect_uri=https://clientapphostaddress/signin-oidc
&scope=openid profile
&response_type=code id_token
&response_mode=form_post
&nonce=63626...n2eNMxA0
```

The Hybrid Flow

Redirection endpoint at client level

```
https://idphostaddress/connect/authorize?
client_id=imagegalleryclient
&redirect_uri=https://clientapphostaddress/signin-oidc
&scope=openid profile
&response_type=code id_token
&response_mode=form_post
&nonce=63626...n2eNMxA0
```

The Hybrid Flow

Requested scopes by the client application

```
https://idphostaddress/connect/authorize?
client_id=imagegalleryclient
&redirect_uri=https://clientapphostaddress/signin-oidc
&scope=openid profile
&response_type=code id_token
&response_mode=form_post
&nonce=63626...n2eNMxA0
```

The Hybrid Flow

The requested response_type determines the flow

Response Type Values

`code` **Authorization Code**	`id_token` **Implicit**	`id_token token` **Implicit**
`code id_token` **Hybrid**	`code token` **Hybrid**	`code id_token token` **Hybrid**

The Hybrid Flow

Client application
(relying party)

IDP

authentication request

authorization endpoint

user authenticates

(user gives consent)


code id_token


code id_token

token is validated

token request (code, clientid, clientsecret)

token endpoint

id_token

id_token

token is validated

The Hybrid Flow

Front channel communication

Information delivered to the browser via URI or Form POST (response_mode)

In OIDC: authorization endpoint

Back channel communication

Server to server communication

In OIDC: token endpoint

Demo

Configuring IdentityServer to Log In with the Hybrid Flow

Demo

Logging in with the Hybrid Flow

Demo

Logging out of Our Web Application

Demo

Logging out of the Identity Provider

Demo

Redirecting After Logging Out

Demo

Returning Additional Claims (Part 1)

```
new Client {
    ClientId = "imagegalleryclient",
    AlwaysIncludeUserClaimsInIdToken = true,
    …
}
```

The UserInfo Endpoint

IdentityServer doesn't include identity claims (except sub) in the identity token, unless we specifically ask for this

The UserInfo Endpoint

Not including the claims in the id_token keeps the token smaller, avoiding URI length restrictions

The UserInfo Endpoint

UserInfo endpoint (IDP level)

- Used by the client application to request additional user claims
- Requires an access token with scopes related to the claims that have to be returned

The Hybrid Flow (UserInfo Endpoint)

Client application
(relying party)

IDP

authentication request

authorization endpoint

user authenticates

(user gives consent)

code id_token

code id_token

token is validated

token request (code, clientid, clientsecret)

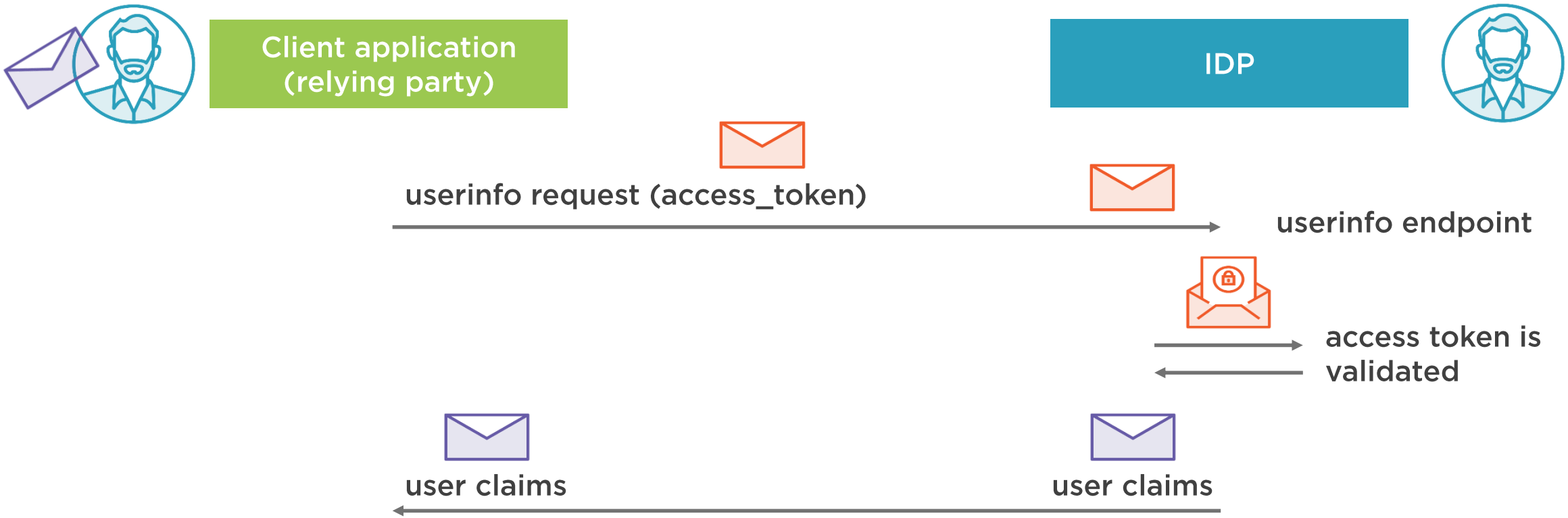
token endpoint

id_token, access_token

id_token, access_token

tokens are validated

The Hybrid Flow (UserInfo Endpoint)



Demo

Returning Additional Claims (Part 2)

```
{
  "sub": "b7539694-97e7-4dfe-84da-b4256e1ff5c7",
  "given_name": "Claire",
  "iss": "https://localhost:44303",
  "aud": "imagegalleryclient",
  …
}
```

Inspecting an Identity Token

Identity tokens are JWTs (Json Web Token)

```
{
  "sub": "b7539694-97e7-4dfe-84da-b4256e1ff5c7",
  "given_name": "Claire",
  "iss": "https://localhost:44303",
  "aud": "imagegalleryclient",
  …
}
```

Inspecting an Identity Token

Subject: the user's identifier

```
{
  "sub": "b7539694-97e7-4dfe-84da-b4256e1ff5c7",
  "given_name": "Claire",
  "iss": "https://localhost:44303",
  "aud": "imagegalleryclient",
  …
}
```

Inspecting an Identity Token

Optional user claims related to the requested scopes

```
{
  "sub": "b7539694-97e7-4dfe-84da-b4256e1ff5c7",
  "given_name": "Claire",
  "iss": "https://localhost:44303",
  "aud": "imagegalleryclient",
  …
}
```

Inspecting an Identity Token

Issuer: the issuer of the identity token

```
{
  "sub": "b7539694-97e7-4dfe-84da-b4256e1ff5c7",
  "given_name": "Claire",
  "iss": "https://localhost:44303",
  "aud": "imagegalleryclient",
  …
}
```

Inspecting an Identity Token

Audience: the intended audience for this token

```
{ …
  "iat": 1490970940,
  "exp": 1490971240,
  "nbf": 1490970940,
  "auth_time": 1490970937,
  …
}
```

Inspecting an Identity Token

Issued At: the time at which the JWT was issued

```
{ …
  "iat": 1490970940,
  "exp": 1490971240,
  "nbf": 1490970940,
  "auth_time": 1490970937,
  …
}
```

Inspecting an Identity Token

Expiration: the expiration time on or after which the identity token must not be accepted for processing

```
{ …
  "iat": 1490970940,
  "exp": 1490971240,
  "nbf": 1490970940,
  "auth_time": 1490970937,
  …
}
```

Inspecting an Identity Token

Not Before: the time before which the identity token must not be accepted for processing

```
{ …
  "iat": 1490970940,
  "exp": 1490971240,
  "nbf": 1490970940,
  "auth_time": 1490970937,
  …
}
```

Inspecting an Identity Token

Authentication Time: the time of the original authentication

```
{ …
  "amr": ["pwd"],
  "nonce": "63…200.ZjMzZ…5YzFlNWNiN2Mw…AtNGYyZi00MzYzNmZh",
  "c_hash": "v1A_h-VQgAvB0-pthVCjJQ",
  "at_hash": "90V_c-P00kdoP-IOERlkdi"
}
```

Inspecting an Identity Token

Authentication Methods References: identifiers for authentication methods

```
{ …
  "amr": ["pwd"],
  "nonce": "63…200.ZjMzZ…5YzFlNWNiN2Mw…AtNGYyZi00MzYzNmZh",
  "c_hash": "v1A_h-VQgAvB0-pthVCjJQ",
  "at_hash": "90V_c-PO0kdoP-IOERlkdi"
}
```

Inspecting an Identity Token

Number only to be used once

```
{ …
  "amr": ["pwd"],
  "nonce": "63…200.ZjMzZ…5YzFlNWNiN2Mw…AtNGYyZi00MzYzNmZh",
  "c_hash": "v1A_h-VQgAvB0-pthVCjJQ",
  "at_hash": "90V_c-P00kdoP-IOERlkdi"
}
```

Inspecting an Identity Token

Code Hash & Access Token Hash: Base64 encoded values of the left-most half of the hash of the octets of the ASCII representation of the code or access token respectively

Summary

Using response_type = code id_token

- Ensures the id_token & code are returned via the front channel
- Allows verifying the id_token first

Front channel communication goes via the browser

Back channel communication is server to server communication

Summary

ClaimsIdentity is created from a validated id_token

Claims can be returned from the UserInfo endpoint to avoid issues with URL length restrictions

When logging out, remember to log out of the IDP if required