OAuth 2.0 and OpenID Connect

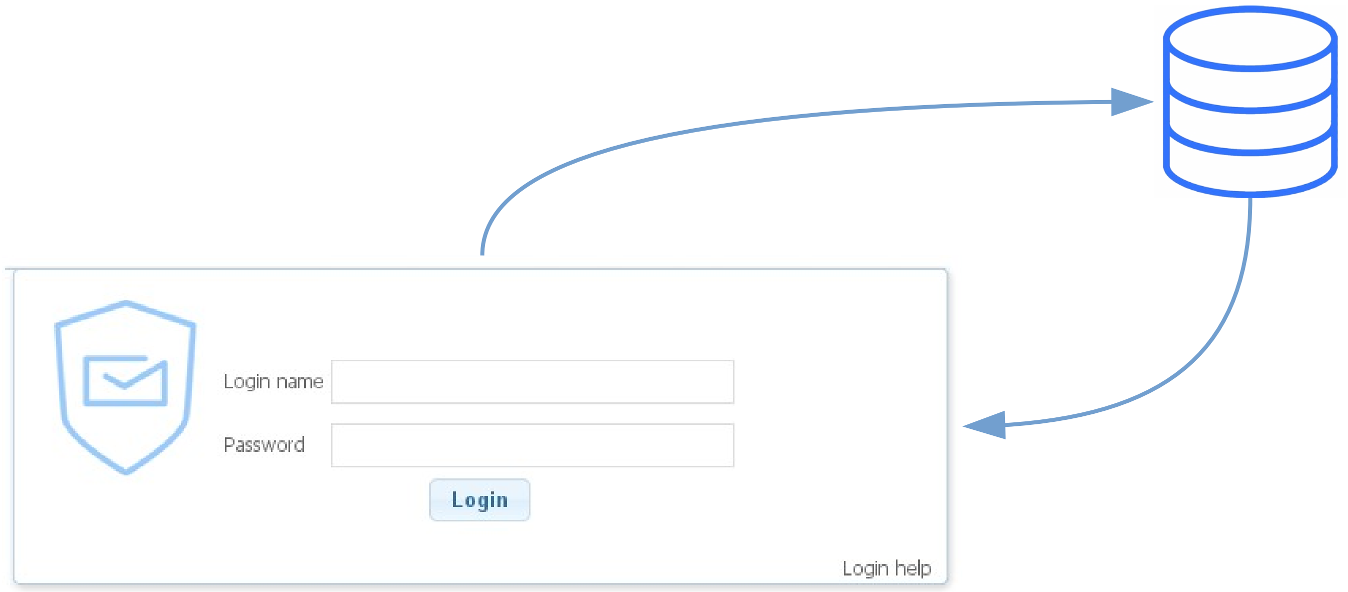
David van der Maas

Agenda

- History and use cases
- What is Oauth and OpenID Connect
- How does it work
- Different flows
- Access Manager configurations

Going back in time

- Simpe web or form authentication



- Hash password
- Verify hash
- Look up user info
- Look up authorization

Downside
- Security
- Maintenance

- `Set-Cookie: sessionid=f00b4r; Max-Age: 3600;`

ngage

Statement: OAuth is hard to comprehend

It’s not but there is some confusion about Oauth

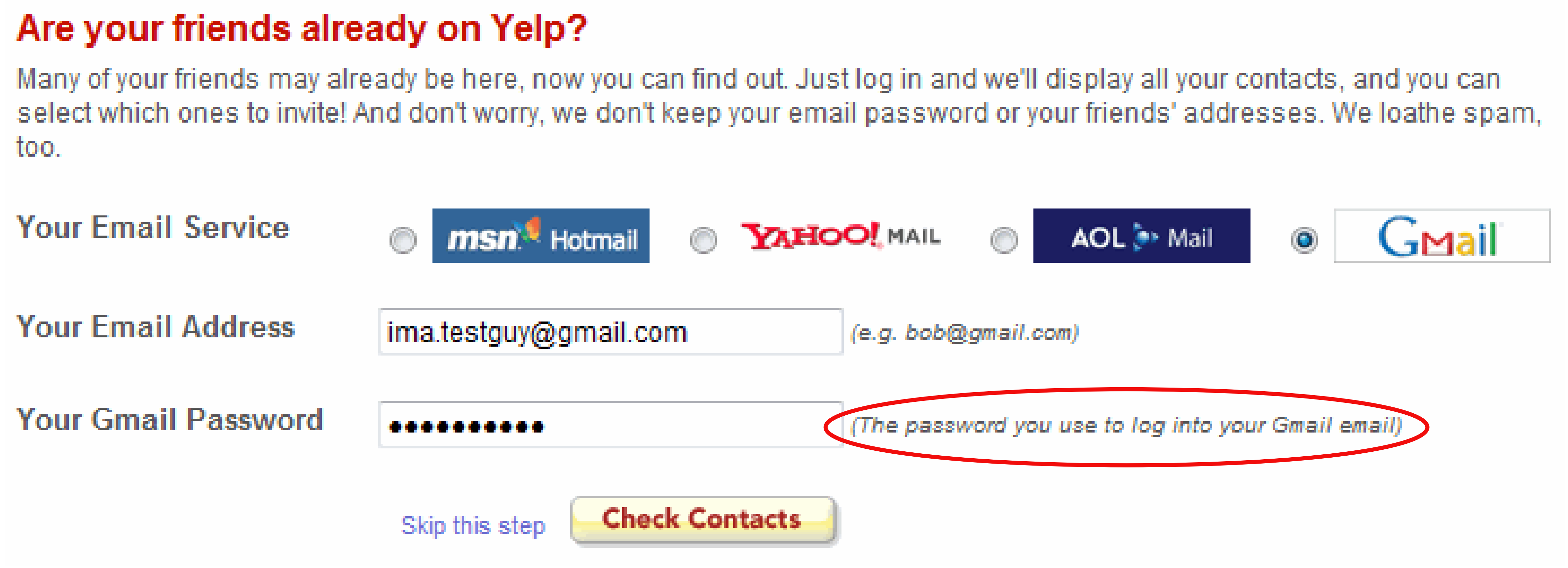
- Terminology & Jargon
- Various and dissimilar advice
- Various possibilities

Going back in time: Use cases

- Local or form login
- Cross security domain SSO (SAML2.0)
  - Still used today, even getting more popular
- Mobile device app login
  - since 2007 with the first iPhone
- Delegated authorization
  - How can an app access my data without my credentials

Going back in time

Delegated authorization as we shouldn't do it



Are your friends already on Yelp?

Many of your friends may already be here, now you can find out. Just log in and we'll display all your contacts, and you can select which ones to invite! And don't worry, we don't keep your email password or your friends' addresses. We loathe spam, too.

Your Email Service	msn Hotmail / YAHOO! MAIL / AOL Mail / [✓ GMail]
Your Email Address	ima.testguy@gmail.com (e.g. bob@gmail.com)
Your Gmail Password	●●●●●●●●●●● (The password you use to log into your Gmail email)

ngage

Challenges

It’s all about securing resources (APIs)

```
GET https:// www.myapp.nl/api/v2/secrets
```

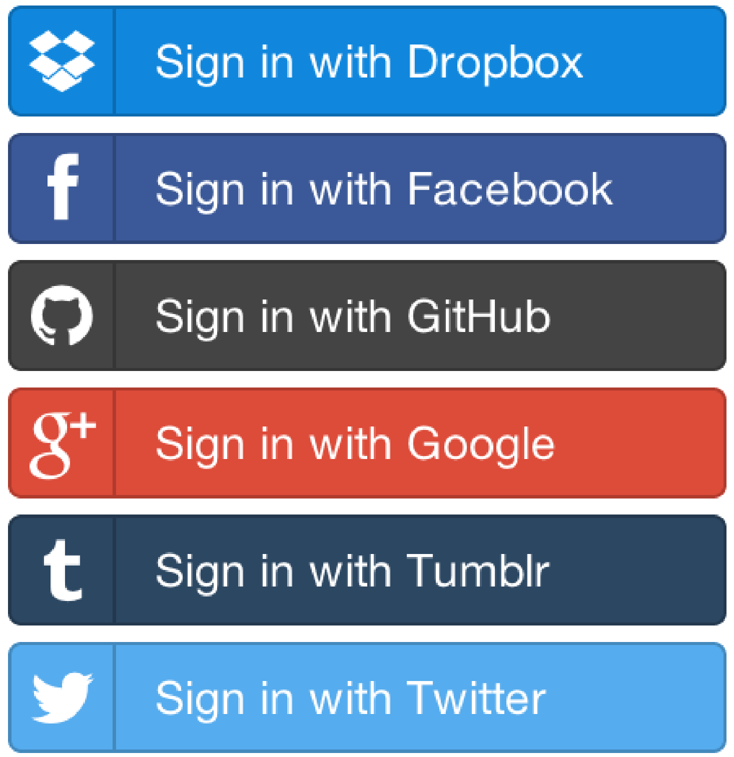
- We want no passwords in files!
  - Who where what , mobile
- Delegate access to act on your behalf
- Selective access
- Revoke access from a central point

OAuth is the solution

Do you use Oauth in your everyday activities

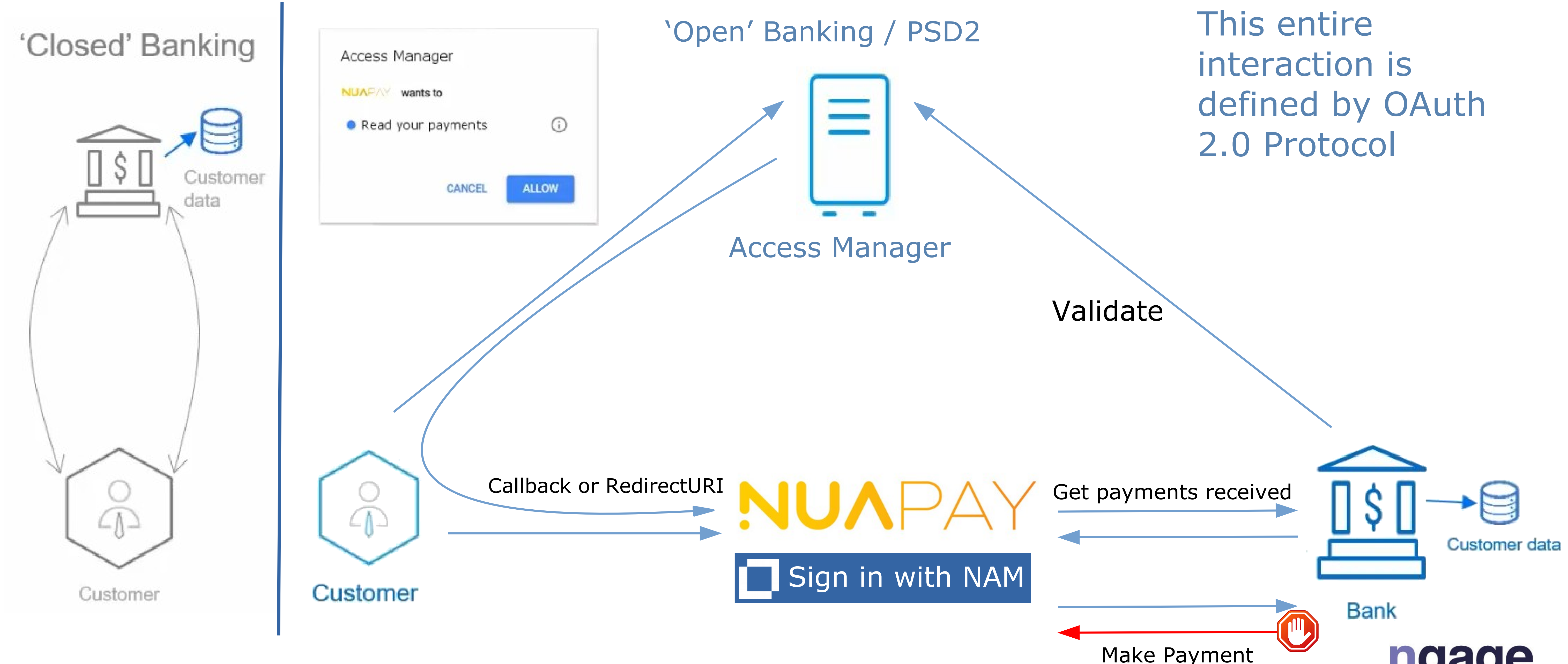
- As a developer
- As a system administrator
- As an end user

Probably you do ….
Oauth powers the billions of social logins that happen every day



ngage

Challenge: Limited Access for Third Party Apps & Web



This entire interaction is defined by OAuth 2.0 Protocol

User wants to delegate selective access to NuaPay

ngage

What is OAuth 2.0

The OAuth 2.0 authorization framework enables a third-party application (nuapay) to obtain limited access to an HTTP server or data (Bank APIs) on behalf of a resource owner (customer).

RFC 6749

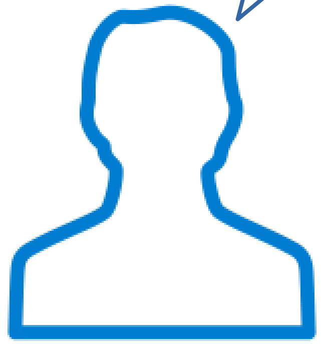
OAuth 2.0 defines how to:

- Delegate access
- Allow Selective access
- Revoke access

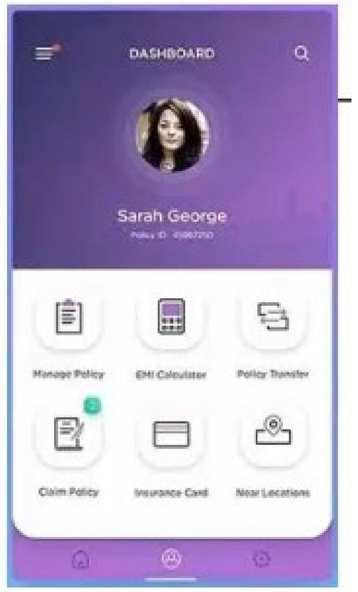
Oauth 2.0 does not how to authenticate a user

Delegated authorization with OAuth 2.0

I trust my insurance provider. I maybe trust my online fitness account and my mobile app and I do want them to have access to my insurance benefits only



Insurance Provider

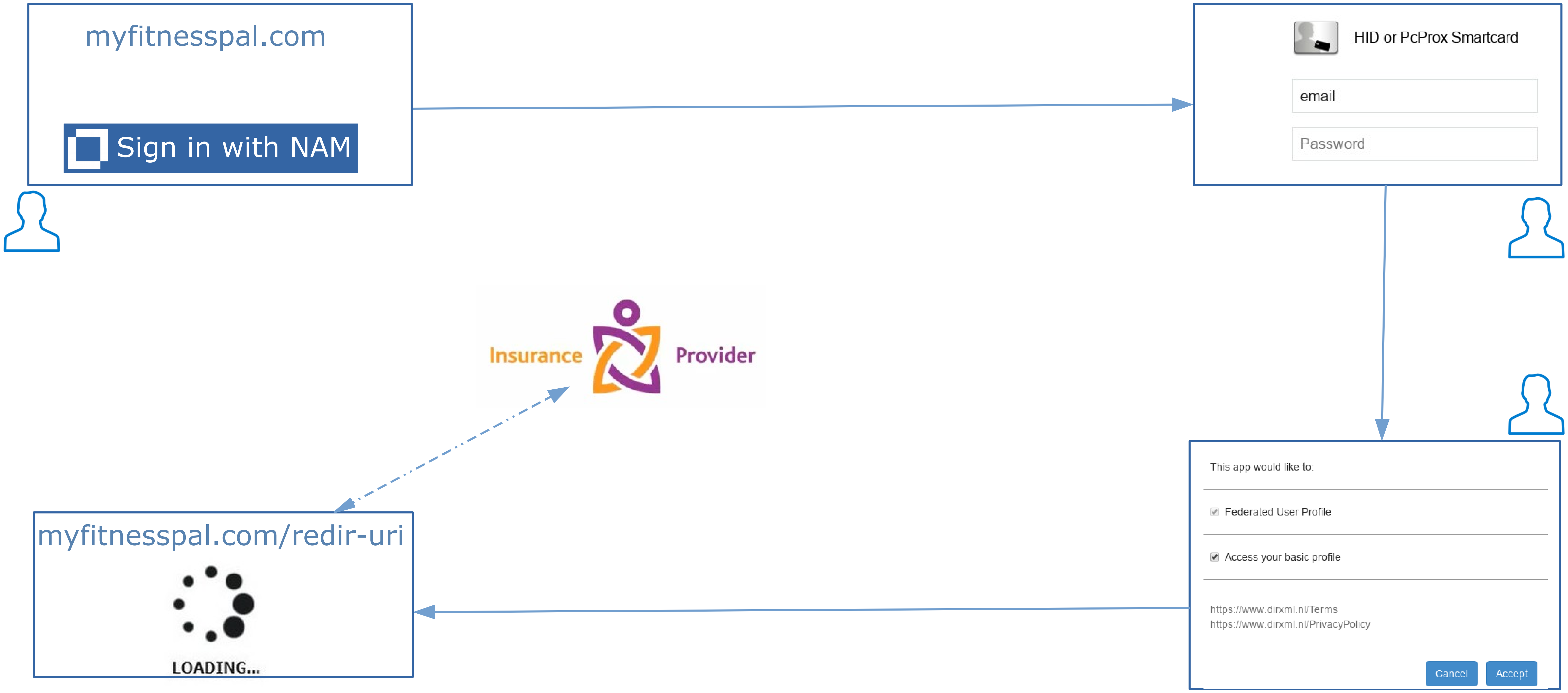


myfitnesspal

myfitnesspal.com

Mobile app

OAuth 2.0 flow for delegated authorization



This app would like to:

- [✓] Federated User Profile
- [✓] Access your basic profile

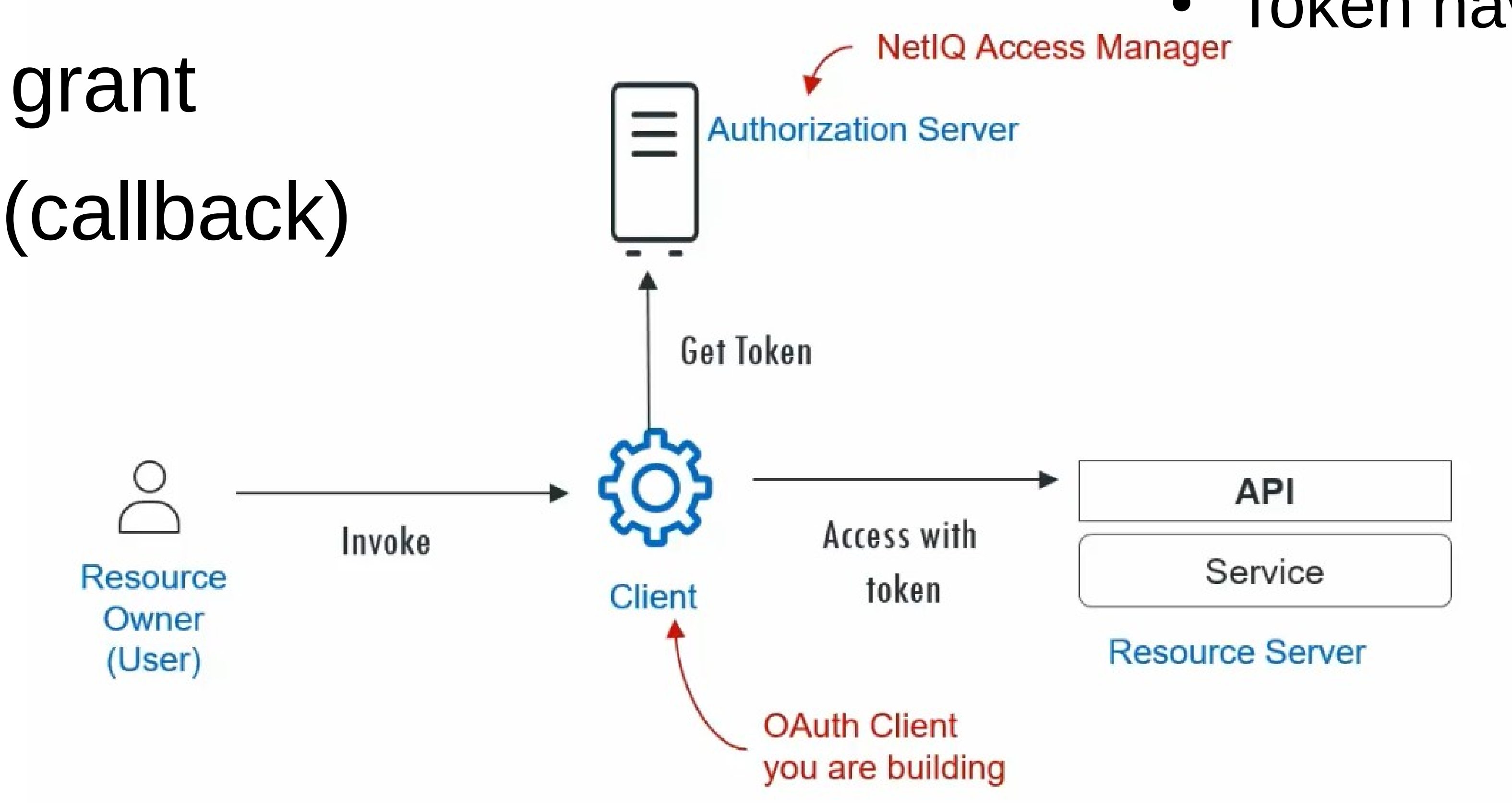
https://www.dirxml.nl/Terms
https://www.dirxml.nl/PrivacyPolicy

OAuth 2.0 terminology (actors)

- Resource owner
- Client
- Authorization server
- Resource server
- Authorization grant
- Redirect URI (callback)
- Access token

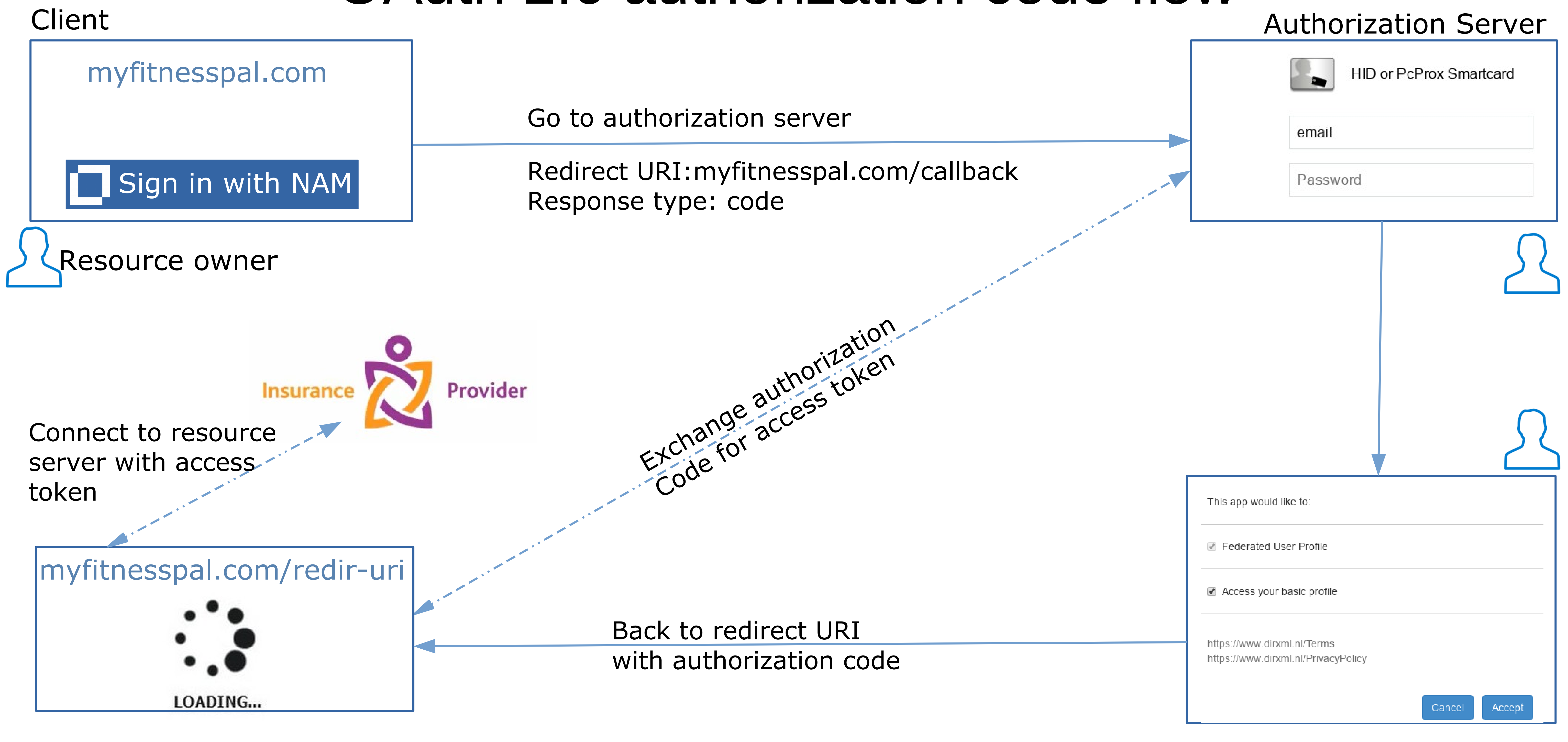
Tokens are the key

- Token have an expiration dat
- Tokens can be renewed
- Tokens can be revoked
- Token have scopes (permissions)



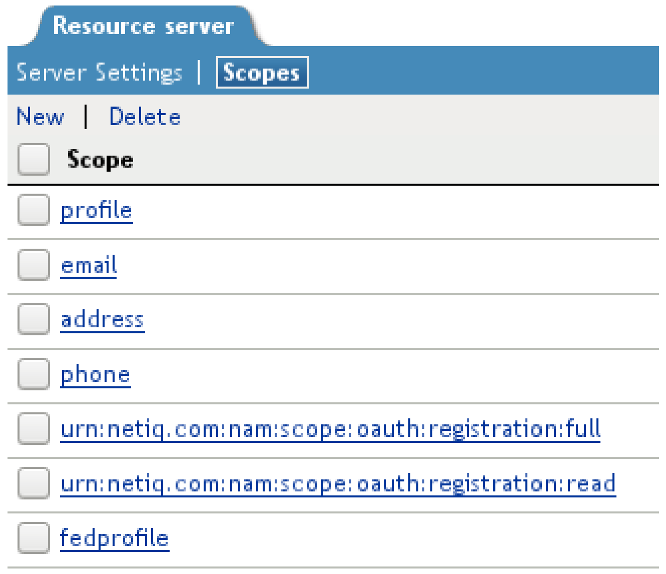
ngage

OAuth 2.0 authorization code flow


Client
myfitnesspal.com
Sign in with NAM
Resource owner
Go to authorization server
Redirect URI:myfitnesspal.com/callback
Response type: code
Authorization Server
HID or PcProx Smartcard
email
Password
Insurance Provider
Connect to resource server with access token
Exchange authorization Code for access token
myfitnesspal.com/redir-uri
LOADING...
Back to redirect URI with authorization code
This app would like to:
Federated User Profile
Access your basic profile
https://www.dirxml.nl/Terms
https://www.dirxml.nl/PrivacyPolicy
Cancel
Accept

More OAuth 2.0 terminology

- Scopes
- Consent

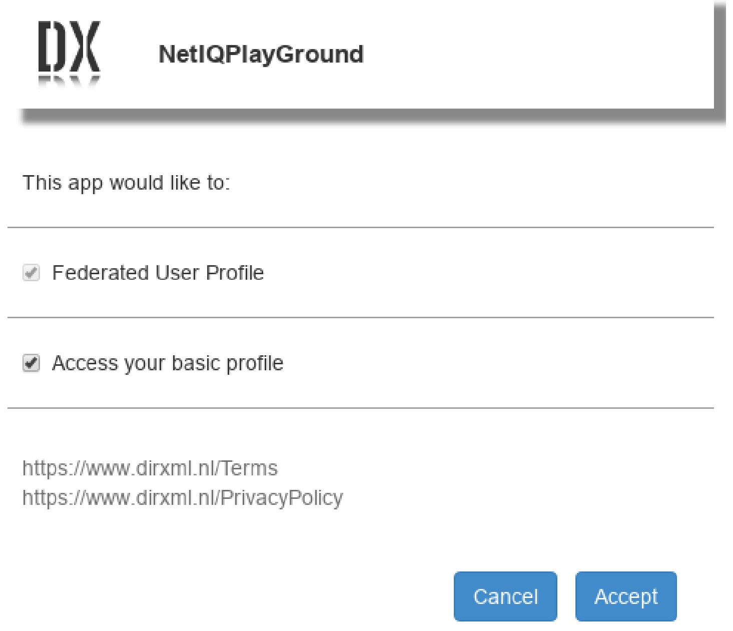


Resource server

Server Settings | Scopes

New | Delete

Scope
profile
email
address
phone
urn:netiq.com:nam:scope:oauth:registration:full
urn:netiq.com:nam:scope:oauth:registration:read
fedprofile

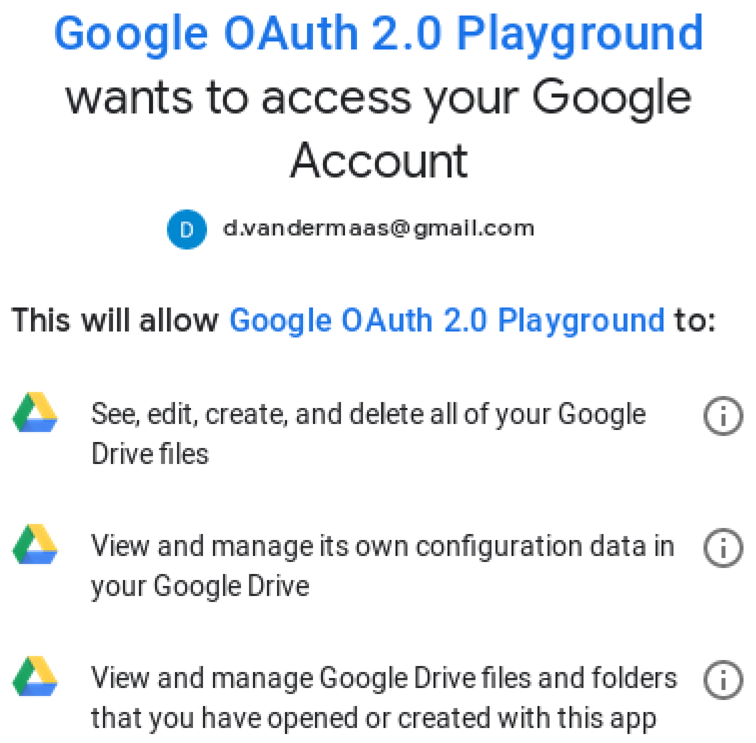


NetIQPlayGround

This app would like to:

- Federated User Profile
- Access your basic profile

https://www.dirxml.nl/Terms
https://www.dirxml.nl/PrivacyPolicy

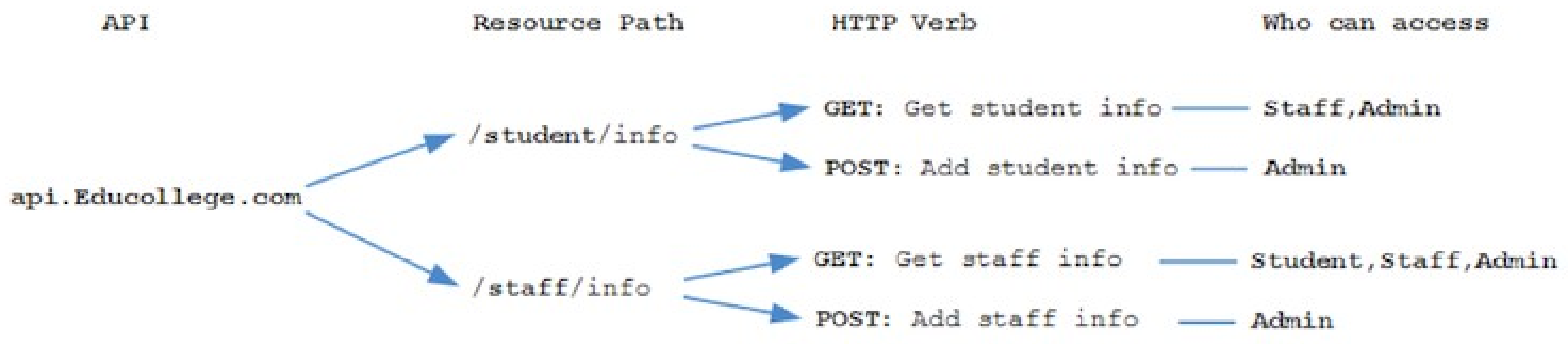


Google OAuth 2.0 Playground wants to access your Google Account

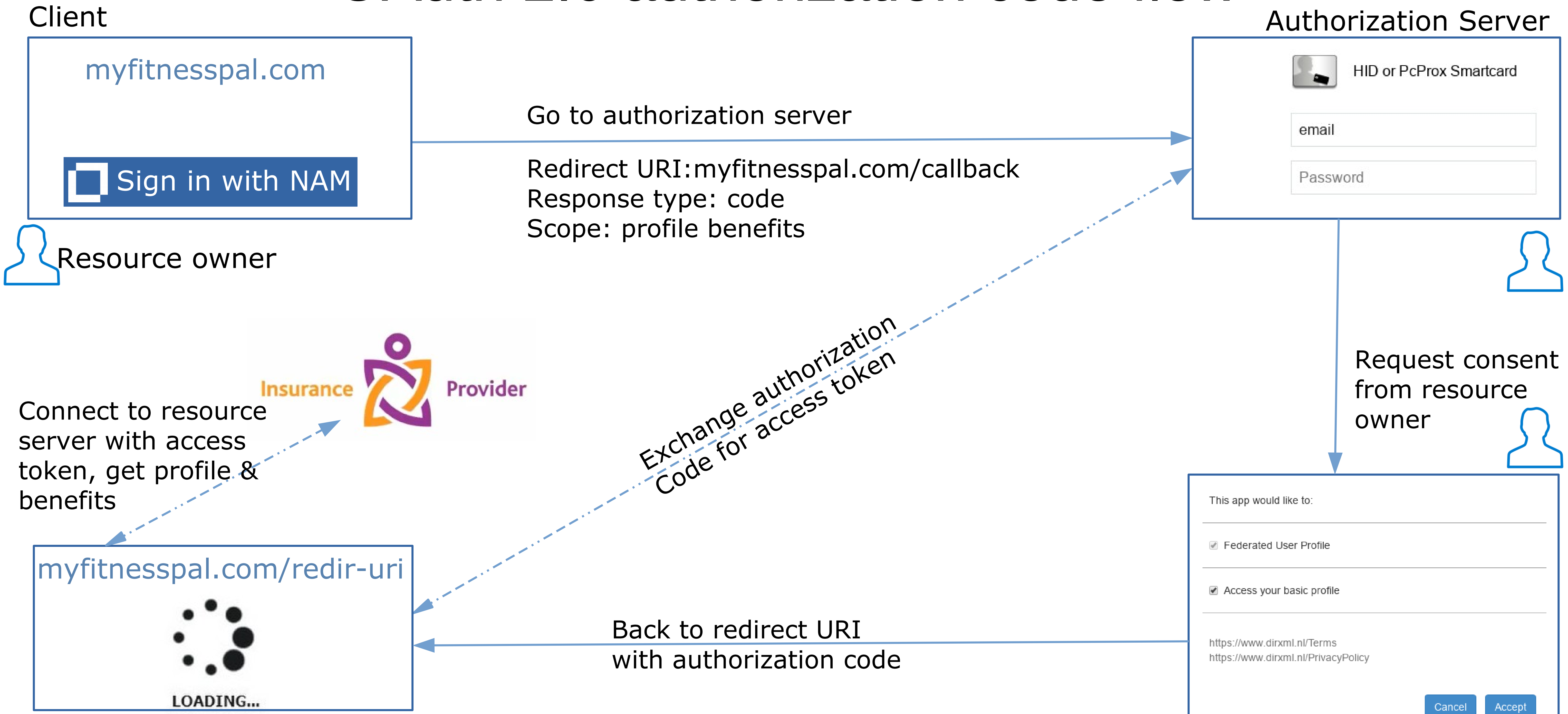
d.vandermaas@gmail.com

This will allow Google OAuth 2.0 Playground to:

- See, edit, create, and delete all of your Google Drive files
- View and manage its own configuration data in your Google Drive
- View and manage Google Drive files and folders that you have opened or created with this app



OAuth 2.0 authorization code flow


Client
myfitnesspal.com
Sign in with NAM
Resource owner
Go to authorization server
Redirect URI:myfitnesspal.com/callback
Response type: code
Scope: profile benefits
Authorization Server
HID or PcProx Smartcard
email
Password
Request consent from resource owner
This app would like to:
Federated User Profile
Access your basic profile
https://www.dirxml.nl/Terms
https://www.dirxml.nl/PrivacyPolicy
Cancel
Accept
Back to redirect URI with authorization code
Exchange authorization Code for access token
Insurance Provider
Connect to resource server with access token, get profile & benefits
myfitnesspal.com/redir-uri
LOADING...

Even more OAuth 2.0 terminology

Client identification at the Authorization server

- Client ID
- Client Secret
- Client needs to be registered

Again more OAuth 2.0 terminology

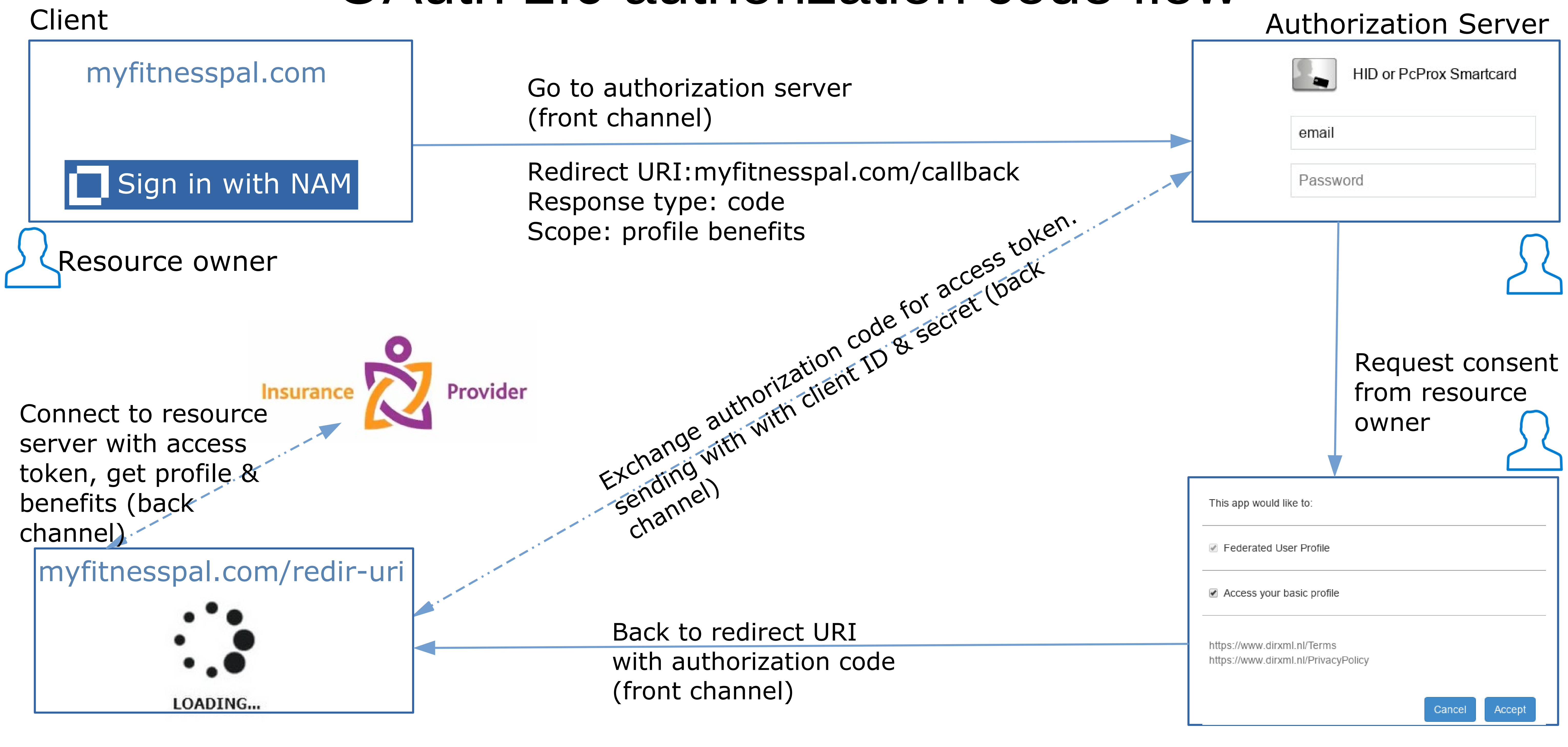
Why exchange authorization code for access token ?

- Back channel (secure channel)
- Front channel (less secure channel)

Where are client ID & secrets stored and where are tokens mantained

- Type of client defines flow (grants)
- Single page client (javascript, angular),
  - no secure storage capabilities for storing client secret
- Web application (jsp on front end, java servlet on back end)
  - has secure storage capabilities for storing client ID and secret in the Java layer

OAuth 2.0 authorization code flow


Client
myfitnesspal.com
Sign in with NAM
Resource owner
Go to authorization server (front channel)
Redirect URI:myfitnesspal.com/callback
Response type: code
Scope: profile benefits
Authorization Server
HID or PcProx Smartcard
email
Password
Request consent from resource owner
This app would like to:
Federated User Profile
Access your basic profile
https://www.dirxml.nl/Terms
https://www.dirxml.nl/PrivacyPolicy
Cancel
Accept
Back to redirect URI with authorization code (front channel)
Exchange authorization code for access token, sending with with client ID & secret (back channel)
Insurance Provider
Connect to resource server with access token, get profile & benefits (back channel)
myfitnesspal.com/redir-uri
LOADING...

Tokens

Access token

- Like a session Will expire
- Contains permissions (scopes)
- Should have short expiration
- Not persisted anywhere

Refresh token

- Like a password
- Exchange for a new Access Token
- Long lived, can be revoked
- Token metadata stored in user attribute

ID token

- Contains user details (claims)
- Part of OpenID Connect
- Mainly used by client
- Not persisted anywghere

- Access token can be exposed on the browser (implicit grant)
- if an access token is compromised there is only a short exposure

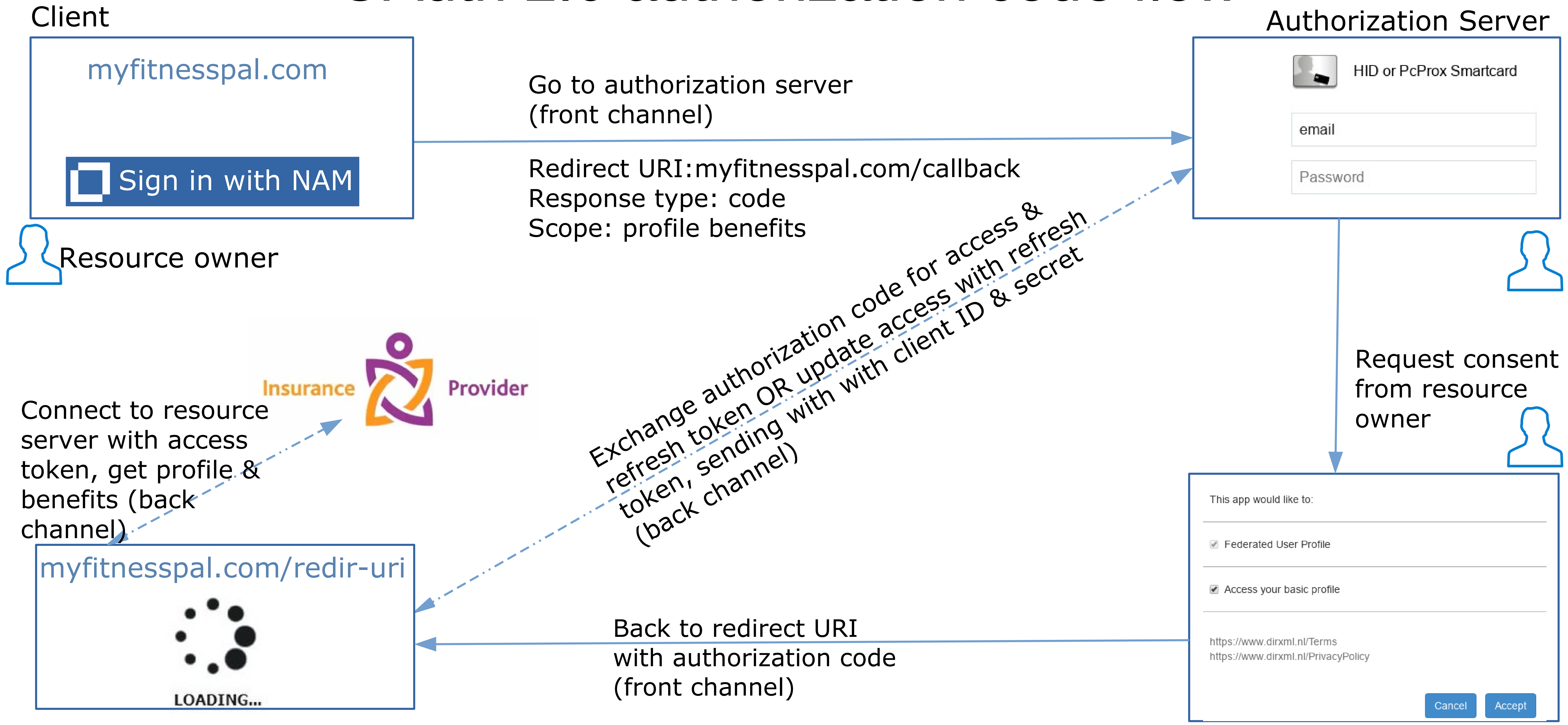
without refresh token

- send API request with access token
- if access token is invalid, fail and ask user to re-authenticate

with refresh token

- send API request with access token
- If access token is invalid, try to update it using refresh token
- if refresh request passes, update the access token and re-send the initial API request
- If refresh request fails, ask user to re-authenticate

OAuth 2.0 authorization code flow



Go to authorization server (front channel)

Redirect URI:myfitnesspal.com/callback
Response type: code
Scope: profile benefits

Request consent from resource owner

This app would like to:

- Federated User Profile
- Access your basic profile

https://www.dirxml.nl/Terms
https://www.dirxml.nl/PrivacyPolicy

Back to redirect URI with authorization code (front channel)

Exchange authorization code for access & refresh token OR update access with refresh token, sending with with client ID & secret (back channel)

Connect to resource server with access token, get profile & benefits (back channel)

Going back in time: Use cases until 2014

• Simple login (OAuth 2.0)	Authentication
• Single sign on across sites (OAuth 2.0)	Authentication
• Mobile app login (OAuth 2.0)	Authentication
• Delegated authorization (OAuth 2.0)	Authorization

OAuth 2.0 and OpenID Connect

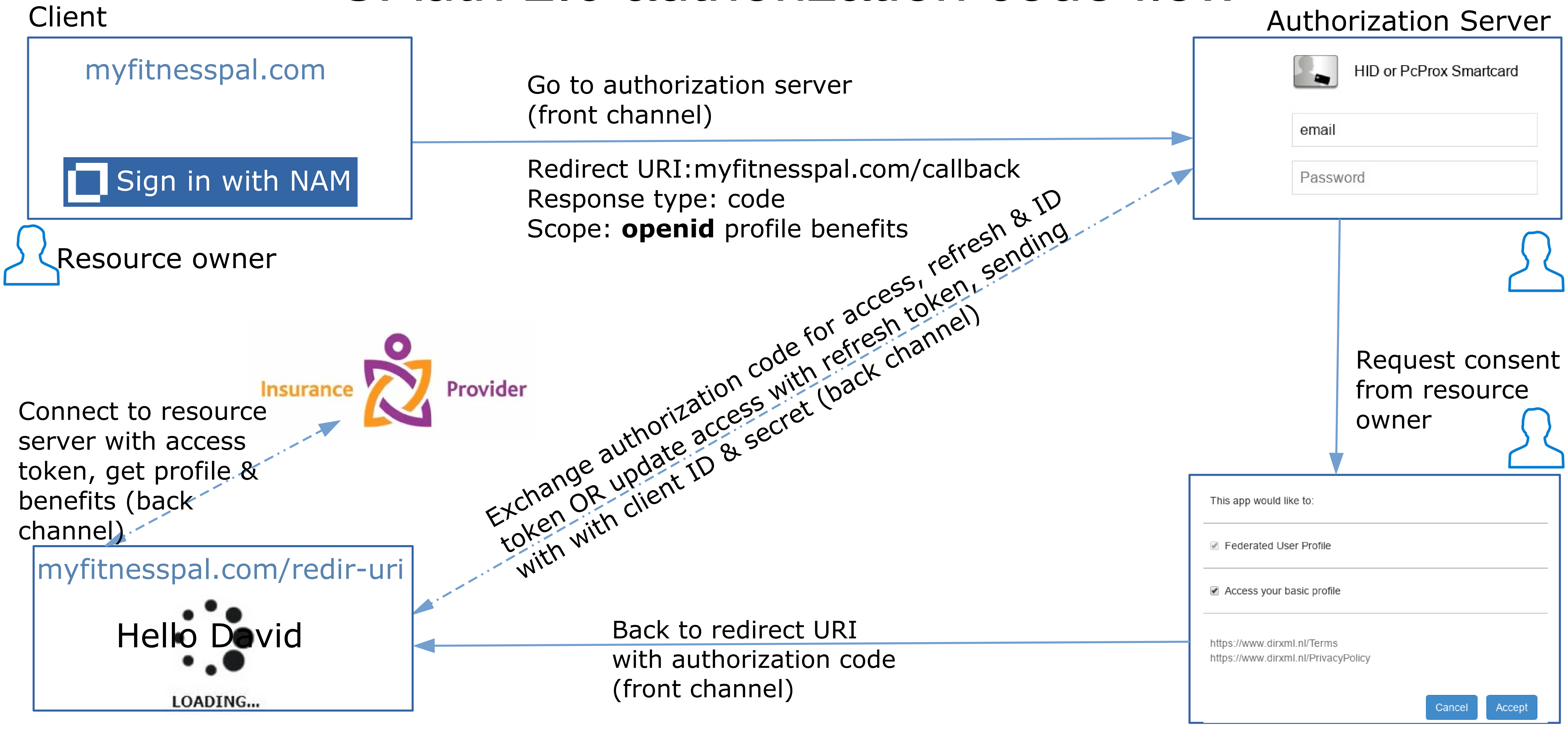
Using OAuth for authentication

- No standards for exchanging user info
- Every deployment is different
- No common set of scope

OpenID Connect is for authentication, OAuth is for authorization

- Not even a seperate protocol, a layer on Oauth
- Provider issues Access & Refresh token AND ID Token
- Standard set of scopes and implementation
- Userinfo endpoint fot getting more information and validate user

OAuth 2.0 authorization code flow



Go to authorization server (front channel)

Redirect URI:myfitnesspal.com/callback
Response type: code
Scope: **openid** profile benefits

Request consent from resource owner

Back to redirect URI with authorization code (front channel)

Exchange authorization code for access, refresh & ID token OR update access with refresh token, sending with with client ID & secret (back channel)

Connect to resource server with access token, get profile & benefits (back channel)

Use cases today

• Simple login (OIDC)	Authentication
• Single sign on across sites (OIDC)	Authentication
• Mobile app login (OIDC)	Authentication
• Delegated authorization (OIDC)	Authorization

OAuth2.0 and OIDC

OAuth 2.0 : Authorization

- Granting access to API
- Getting access to user data in other systems

OpenID Connect : Authentication

- User login
- Making accounts available in other systems

Authorization Code flow: Start

```
https://login.dirxml.nl/nidp/oauth/nam/authz?
   scope=profile+openid+fedprofile&
   response_type=code&
   redirect_uri=https://myapp.webapps.com/netiq-playground/oauth2client&
   client_id=002eb3d9-e9af-4370-bb49-00ae9d87f5b3
```

Authorization Code flow: Callback

```
https://myapp.webapps.com/netiq-playground/oauth2client?
     error=access_denied&
     error_description=user has denied the grants to client

https://myapp.webapps.com/netiq-playground/oauth2client?
     code=eyJhbGciOiJBMTI4S1ciLCJlbmMiOiJBMTI4R0NNIiwi&
     scope=fedprofile+profile
```

ngage

Authorization Code flow: Request Access Token

```
POST https://login.dirxml.nl/nidp/oauth/nam/token
Content-Type: application/x-www-form-urlencoded

  code=eyJhbGciOiJBMTI4S1c&
  grant_type=authorization_code&
  client_secret=_aM6OUG8nG5v_NWTdddThisIsFakeBcXKlSJlOeVUS&
  client_id=002eb3d9-e9af-4370-bb49-00ae9d87f5b3&
  redirect_uri=https://myapp.webapps.com/netiq-playground/oauth2client
```

Authorization Code flow: Request Access Token

```
POST https://login.dirxml.nl/nidp/oauth/nam/token
Content-Type: application/x-www-form-urlencoded

  code=eyJhbGciOiJBMTI4S1c&
  grant_type=grant_type=refresh_toke&
  refresh_token=eyJraadfcSDEaWQssssiOi&
  client_secret=_aM6OsssUG8nG5v_NWTdddThisIsFakeBcXKlSJlOeVUS&
  client_id=002eb3d9-e9af-4370-bb49-00ae9d87f5b3&
  redirect_uri=https://myapp.webapps.com/netiq-playground/oauth2client
```

Authorization Code flow: Get Access Token

```
{
  “access_token”: “xnrhmhZuZRxX5AlsoFake48SVefE6peJf”,
  “expires_in”: 2480,
  “token_type”: “Bearer”,
}
```

Authorization Code flow: Using Access Token

```
GET https://myapp.webapps.com/api/oauth/v4/benefits
Authorization: Bearer xnrhmhZuZRxX5AlsoFake48SVefE6peJf
```

Endpoint validates token

Endpoint uses scope for Authorization

ngage

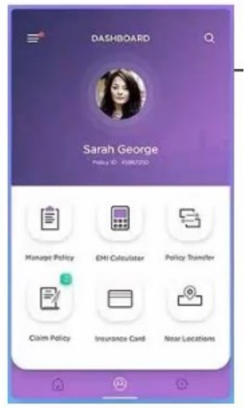
Calling userinfo endpoint

```
GET login.dirxml.nl/nidp/oauth/nam/userinfo
Authorization: Bearer kiuniIuyIYYGysiIUSuhinIUS

200 OK
Content-Type: application/json
{
    “sub”: “david@dirxml.nl”
    “name”: “david van der Maas”
    “shoesize”: “42”
}
```

Demo

Different OAuth 2.0 flows



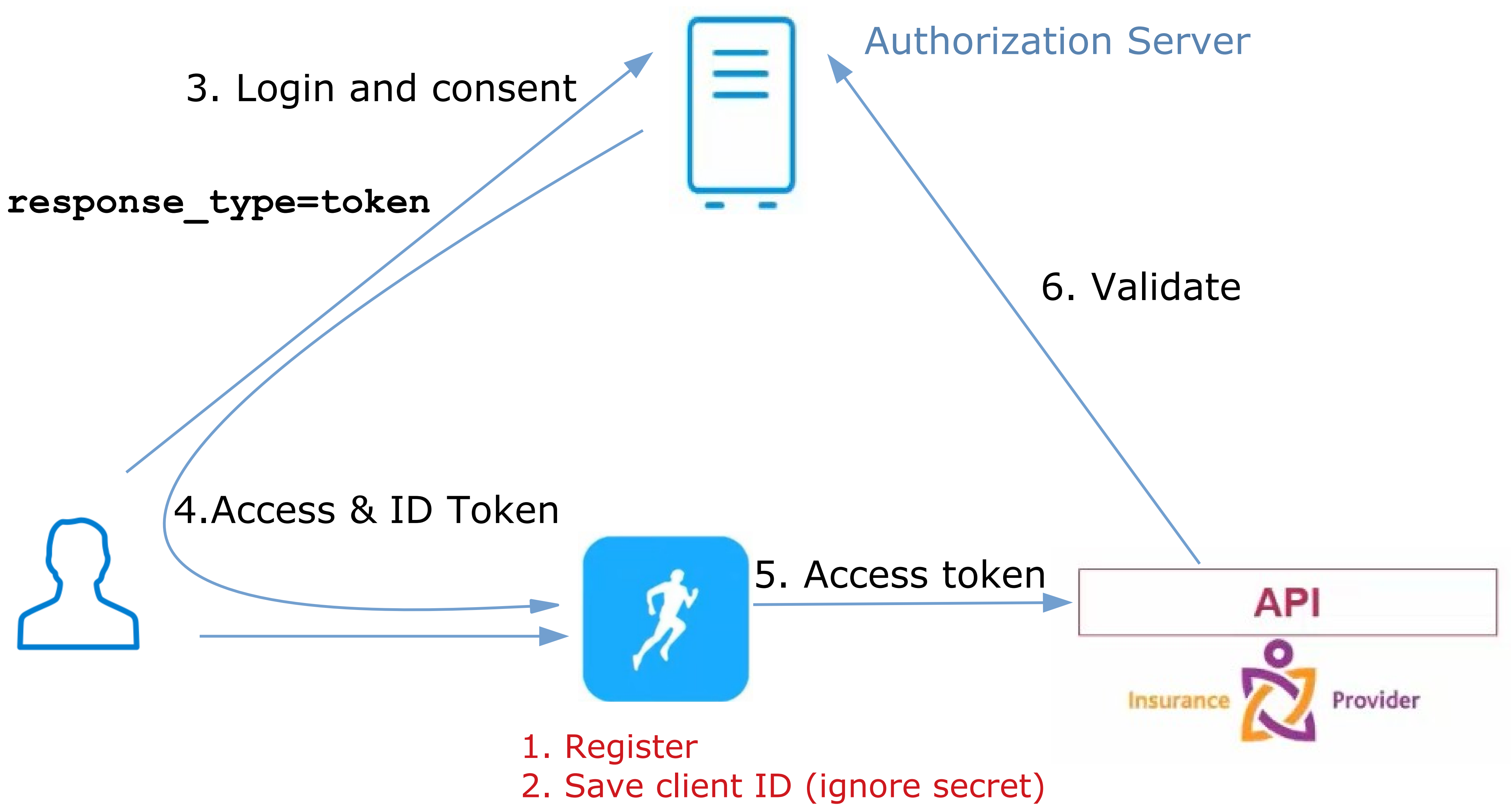
myfitnesspal.com

Mobile app

- Authorization code
- Authorization code with PKCE
- Implicit
- Resource Owner Password credentials
- Client credentials
- SAML 2.0 Bearer grant

ngage

Oauth 2.0 flow: Implicit Grant



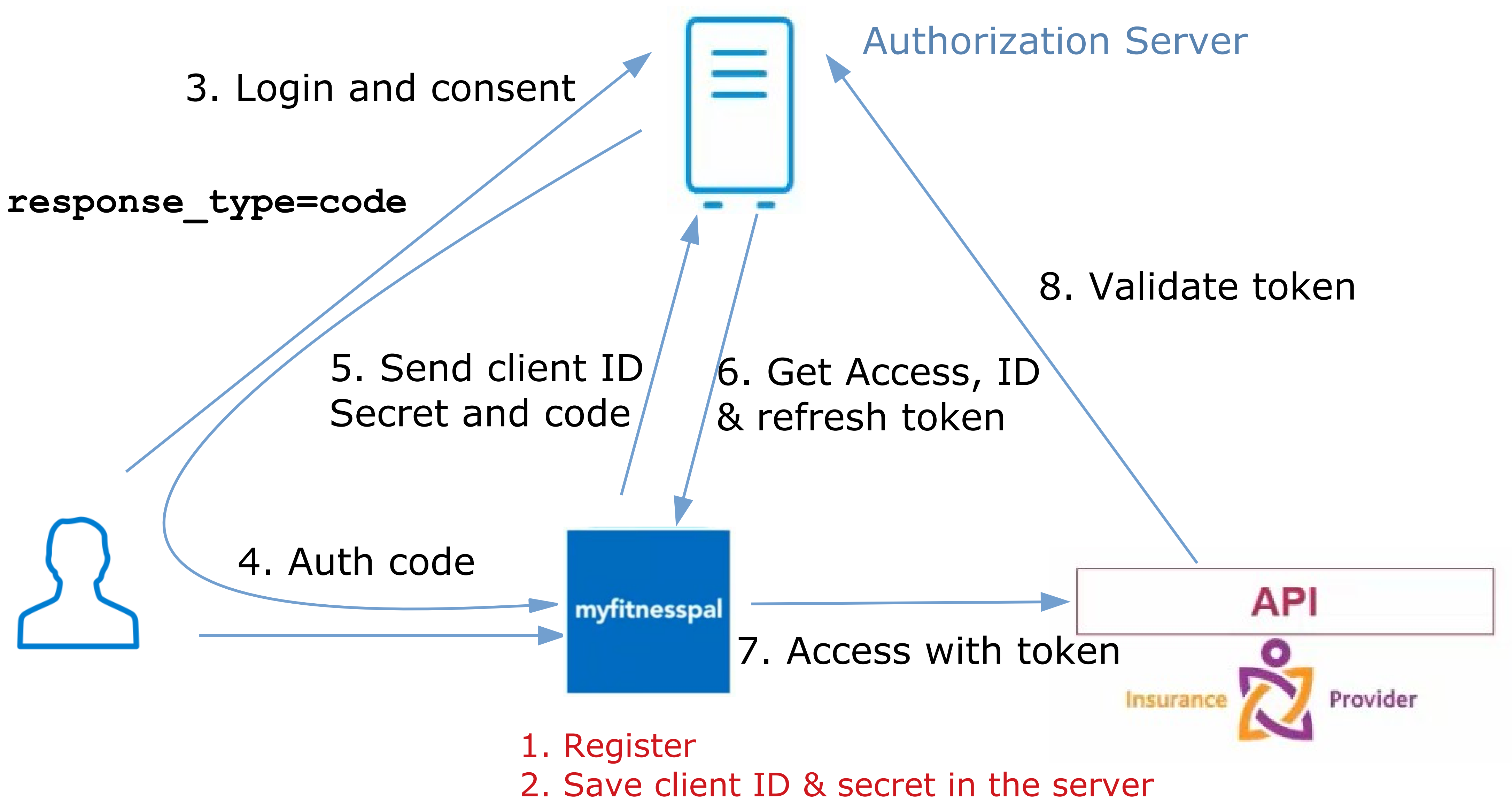
Implicit grant

- Mainly for public clients
- Front channel only
- Simple to implement
- Access Token sent thru browser
- No refresh token
- Cannot securely store client secret

Suitable for

- Single Page Applications
- Mobile Apps
- Desktop Apps

Oauth 2.0 flow: Authorization Code Grant



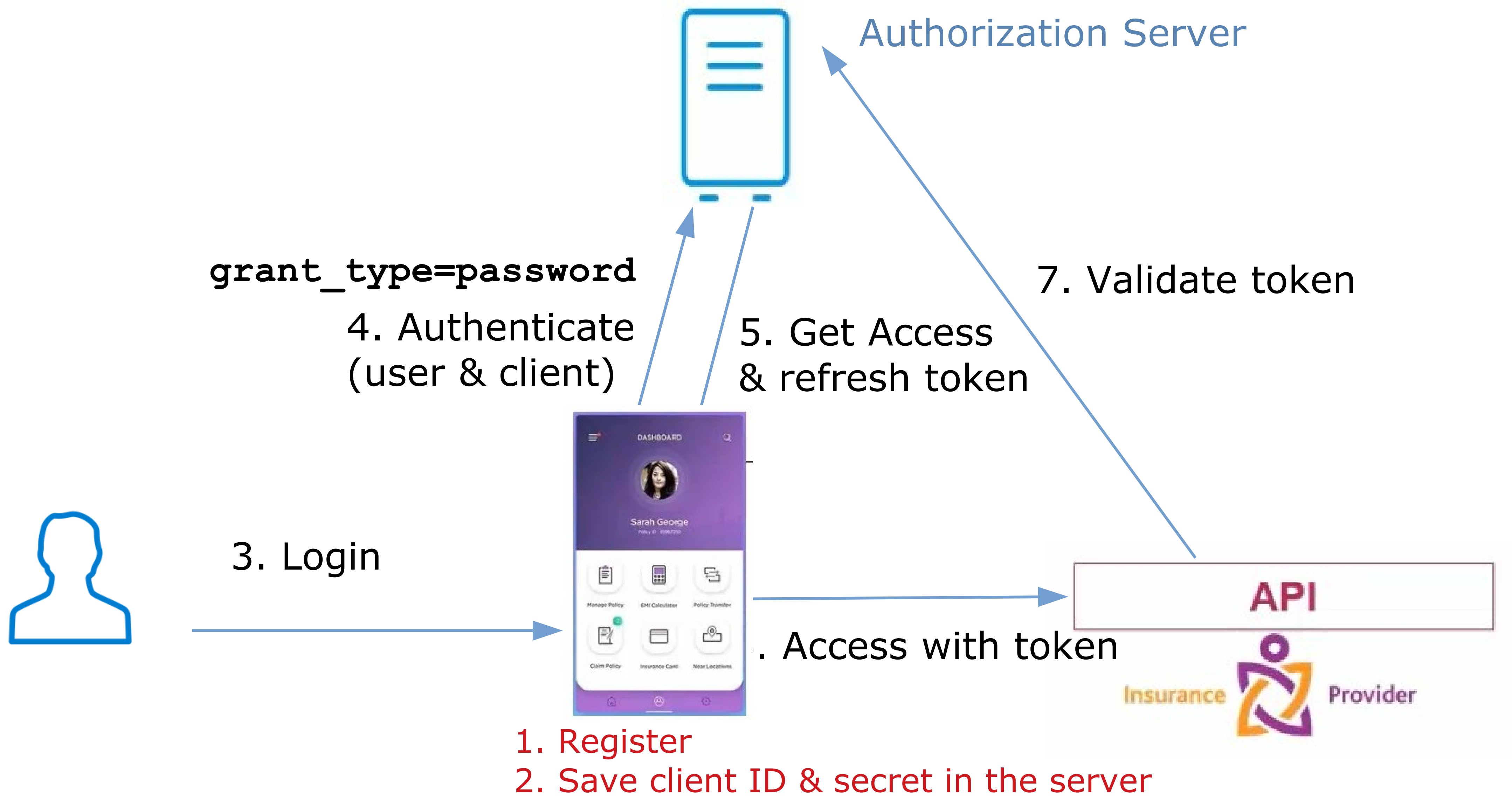
Authorization code grant

- Confidential clients
- Front and back channel
- Clients must handle browser redirects & HTTPS
- Client secret & ID securely stored

Suitable for

- Web applications
- Mobile Apps
- Native apps (with PKCE, Proof Key for Code Exchange)

Oauth 2.0 flow: Resource owner password credentials



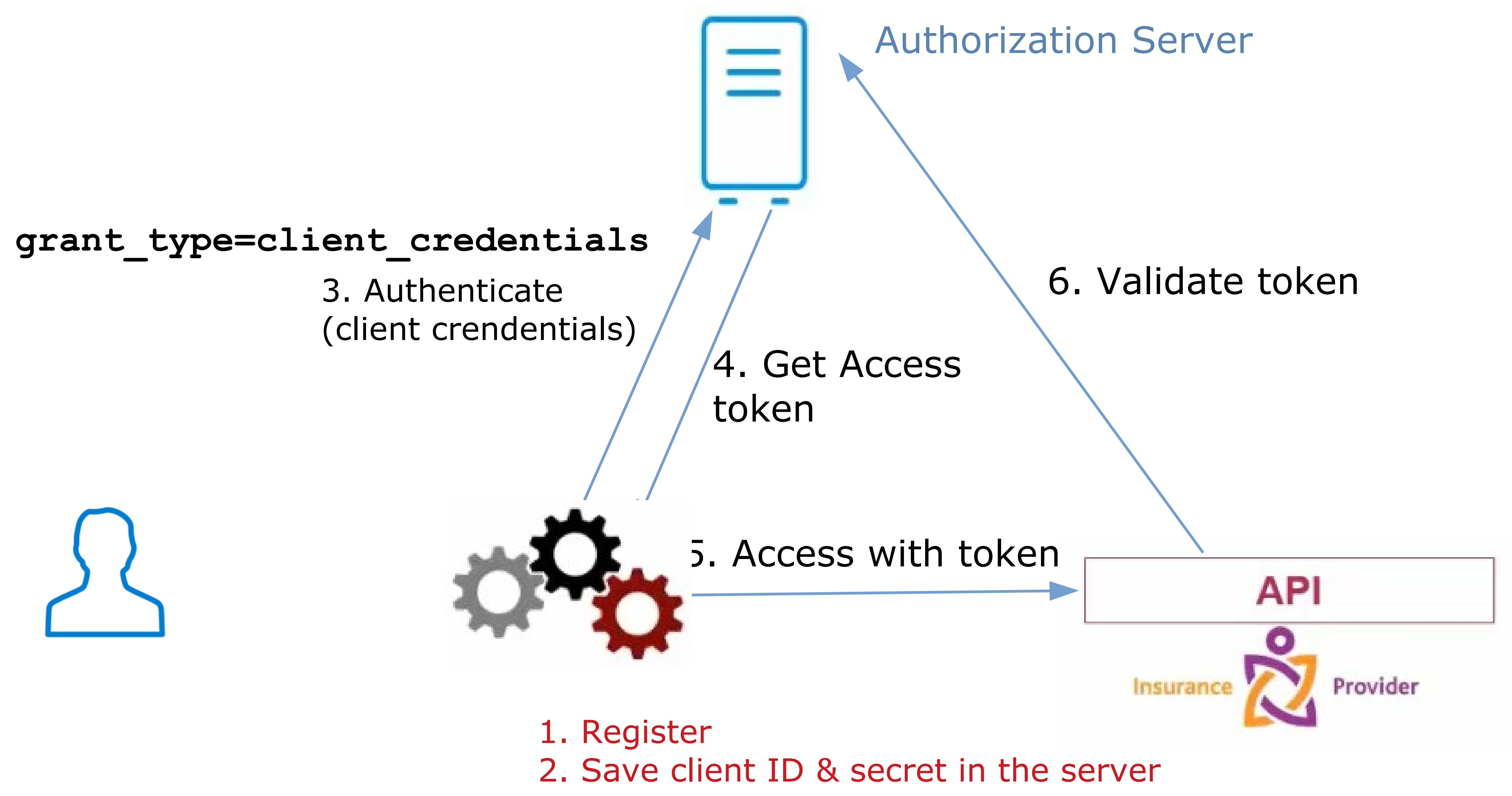
Authorization code grant

- Trusted clients
- Back channel only
- Clients is trusted with name / password (!)
- Client & User secret & ID securely stored
- Refresh token supported

Suitable for

- Highly trusted apps
- Native apps (to migrate)

Oauth 2.0 flow: Client credentials



Authorization code grant

- Machine to machine
- No user involved
- Back channel only
- Clients is trusted with name / password (!)
- Refresh token not supported

Suitable for

- Headless clients
- Microservices
- APIs – Batch processing

Registering Oauth Client Application

Administration Console

Access Manager Administrator can register and manage all OAuth clients.

IDP User Portal

OAuth Developer : Register and manage their own OAuth clients.

OAuth Admin : Register and manage all OAuth clients.

REST API

Can be used to create multiple clients at a time. Oauth Developer Docs contains all details.

ngage

OAuth Support in Access Manager

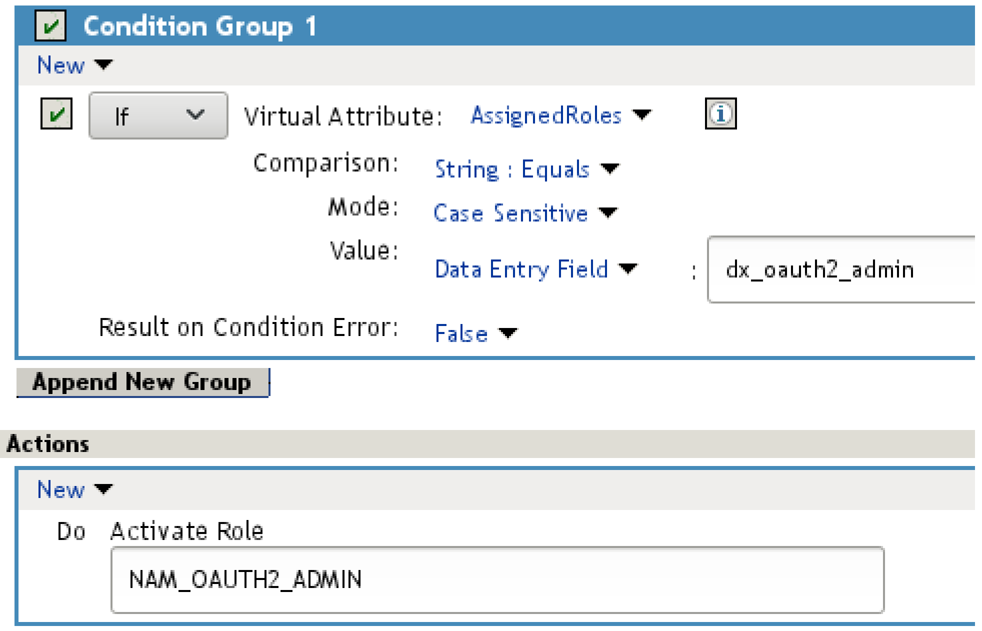
- The IDP functions as an OAuth 2.0 authorization server
  - IDP can authenticate resource owners, obtain their authorization and issue access token to client applications
- Supports
  - Authorization code grants
  - Implicit grants
  - Resource Owner Credentials grant
  - Client credential grants
- OpenID Connect implements a single sign-on protocol on top of the OAuth authorization process.

OAuth Support in Access Manager (Cont.)

- Validate OAuth access tokens without redirection
  - Provides ability to convert legacy applications to OAuth flow
- Access Gateway supports access tokens from x-Access IDP
  - Better integration with x-Access
  - SSO to IDM components (OSP)
- REST APIs for client registration and management

OAuth Developer

- Required IDP jobs:
- NAM_OAUTH2_ADMIN
  - Register and manage all OAuth clients in IDP Portal
- NAM_OAUTH2_DEVELOPER
  - Register and manage their own OAuth clients


Condition Group 1

New

If Virtual Attribute: AssignedRoles

Comparison: String : Equals

Mode: Case Sensitive

Value: Data Entry Field : dx_oauth2_admin

Result on Condition Error: False

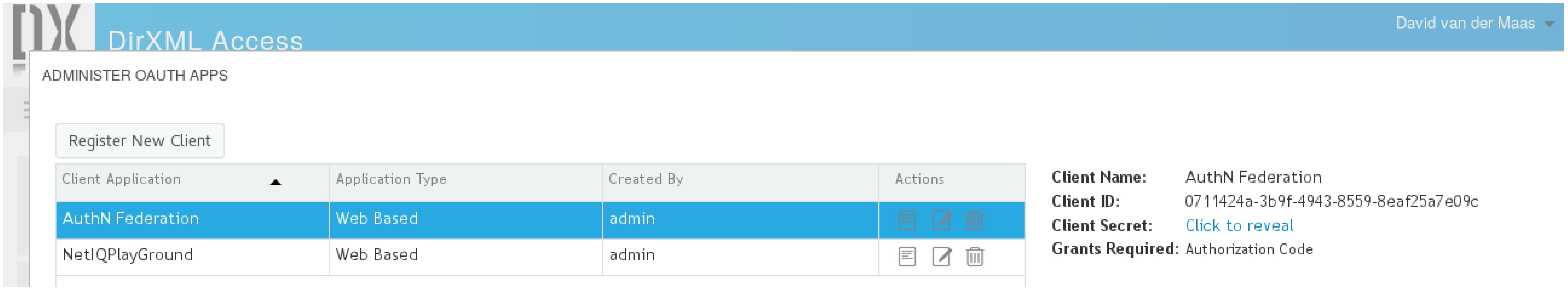
Append New Group

Actions

New

Do Activate Role

NAM_OAUTH2_ADMIN


DirXML Access

David van der Maas

ADMINISTER OAUTH APPS

Register New Client

Client Application	Application Type	Created By	Actions
AuthN Federation	Web Based	admin	
NetIQPlayGround	Web Based	admin	

Client Name:	AuthN Federation
Client ID:	0711424a-3b9f-4943-8559-8eaf25a7e09c
Client Secret:	Click to reveal
Grants Required:	Authorization Code

Demo

ngage

OAuth	OpenID Connect	SAML	WS-* Family
An open protocol to allow secure authorization in a simple and standard method from web, mobile and desktop applications. Provides API authorization between applications.	Provides single sign-on (SSO) layer on top of the OAuth protocol for consumers.	An XML-based open standard data format for exchanging authentication and authorization data between an identity provider and a service provider. Encompasses profiles, bindings and constructs to achieve SSO, federation, and identity management.	Allows secure identity propagation and token exchange between Web services. Enables applications to construct trusted SOAP message exchanges.
OAuth tokens can be binary, JSON, or SAML	Uses JSON tokens	Deals with XML as the data construct or token format.	Uses Request Security Token (RST) and Request Security Token Response (RSTR)
Uses HTTP exclusively	Uses HTTP exclusively	No restriction on the transport format. You can use SOAP, JMS, or any transport you want to use to send SAML tokens or messages.	No restriction on the transport format. You can use SOAP, JMS, or any transport you want to use to send SAML tokens or messages.
Designed for use with applications on the Internet.	Designed for use with applications on the Internet.	Used in enterprise SSO scenarios.	Used in enterprise SSO scenarios.

Q
&
A

www.microfocus.com