Presented by Cal Heldenbrand

- Web Operations at FBS
- cal@FBSData.com
What does it do?

- Protocol that defines a secure OAuth2 implementation
- Provides an identity layer on top of OAuth2
  - In the form of an ID Token
- Adds Claims (profile info) to an identity
  - Via a UserInfo API endpoint
- SSO / SLO
- Creates and uses OAuth2 code/access/refresh tokens
- “Mobile friendly”
Who uses it?

- Google
- Amazon
- Microsoft
- IBM
- PayPal
- eBay
- Salesforce
- Ping Identity
- (Future)
- Yahoo
- AOL
- Facebook
OIDC Philosophy

- Keep Simple Things Simple
  - Everything is in JSON
  - Build only the features you need
  - Features are detected using a discovery service
  - Adding a new Provider is only a few lines of code

- Let Complex Things Be Possible
  - Encrypted ID Tokens
  - ID Token issued to multiple apps
An ID Token is like...

**Identification Card**

Issuer: https://sparkplatform.com
Subject: 2008103015492549172
Audience: bi2b94j6evmocua586ln0bh31
Issued At: 1433435400
Expires: 1433521800
Nonce: dxe9v2

Name: Ned Flanders
NRDS ID: 186879823
Email: ned@simpsons.com
And an Access Token should be...

- Access delegation ONLY (consent)
- Not tied to an identity
- Not for federation
- Not for authorization
  ... wait, what!?

Source: API Security: Deep Dive into OAuth and OpenID Connect
An ID Token

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczovL3NwYXJrcGxhdGZvcm0uY29tIiwic3ViIjoiMjAwODEwMzAxNTQ5MjU0OTE3MiIsImF1ZCI6ImJpMmI5NGo2ZXZtb2N1YTU4NmxuMGJoM2wiLCJleHAiOjE0MzM1MjE4MzAsImlhdCI6MTQzMzQzNTQwMCwibm9uZSI6InNwYXJrcGxhdGZvcm0uY29tIiwiaWQiOjQ4MDIzIn0.Zpe4jBmqMy7zyisPdeRFLr8loX2bDzqQmKEEb2zC2u0du4CjzKDF5mE5FDsiyCytUppt_xAPMXWkJCqmptxif7202EAM6qiCszMfcRH-DjTeKq8MBKbsWncm68xasQfsHfyh5vuJA6rXGU6AP4kash0xX9Rk2xF_JQXkc-QVrybPpwM83Cx9xVkJQrLQ0Zyh7L33uc5Rjyc-DH5dfsgquALnGHO_mjyh1P9p0_AthLW__3usfmYGMIojcfJc2VxPE840BzIG0YNmCPbdvs_bu88wMpwzncDZti8BKUdqS7YYtPc1Lc3PCX0-49o3mCktZd9YxI61WS1XjbsRw
An ID Token Decoded

Header
{
  "typ": "JWT",
  "alg": "RS256"
}

Claims
{
  "iss": "https://sparkplatform.com",
  "sub": "2008103015492549172",
  "aud": "bi2b94j6evmocua586ln0bh31",
  "exp": 1433521800,
  "iat": 1433435400,
  "nonce": "dxe9v2",
  "name": "Ned Flanders",
  "email": "ned@simpsons.com",
  "MemberNrdsId": "186879823",
  "at_hash": "tq7zVbs_DdGzS9o-iOa_VA"
}

Checksum
{
  Zpe4jBmqMy7zyiSPdeRFLr ...
  49o3mCktZd9Yx161WS1XjbsRw
}
OpenID Authentication Flows

- **Authorization Code Flow**
  - Just like the current Web API OAuth2

- **Implicit Flow**
  - ID Token, Access Token given to the client

- **Hybrid Flow**
  - Combo of Implicit and Authorization Code
  - Designed with native mobile apps in mind
  - Reduces your carbon footprint?
Discovery Service

curl https://openidp.fbsdata.com/.well-known/openid-configuration

curl https://accounts.google.com/.well-known/openid-configuration

- Client Developers specify a Base URL
- Client library appends static path
- Providers could use a static JSON file
  - Rarely changes
Discovery Response

{
  "issuer" : "https://openidp.fbsdata.com",
  "response_types_supported": [ "code", "token", "id_token",
                              "code token", "code id_token", "id_token token",
                              "code id_token token"],

  "authorization_endpoint" : "https://openidp.fbsdata.com/authorize",

  "token_endpoint" : "https://openidp.fbsdata.com/token",

  "userinfo_endpoint" : "https://openidp.fbsdata.com/userinfo",


  "claims_supported" : [ "sub", "iss", "name", "given_name",
                        "family_name", "middle_name", "preferred_username", "website",
                        "address", "phone_number", "MemberMlsId",
                        "OfficeKey","MemberNrdsId"
  }
}
UserInfo Endpoint

- Protected by Access Token
- Returns JSON of Claims
- Extended info that can't fit in an ID Token
The Core Specification

- That's it for the core specification!
  - ID Tokens (required)
  - Discovery Service (required)
  - UserInfo Endpoint (optional)

- Implement just two pieces for a certified Provider
  - Or Relying Party (client)
Compliance Testing

- 3rd party tools
  - oictest
  - pyoidc

- OpenID Foundation Certification Program
  - General availability in Jan 2016
  - Open Source Web UI
  - http://openid.net/certification
Demo: TestFormVendor.com

Test Form Vendor - An OpenID Connect Demo

Check the response_types desired

- ID Token (id_token)
- OAuth2 Authorization Code (code)
- OAuth2 Access Token (token)

OpenID Connect Flow

Hybrid

Sign In with SparkPlatform

Sign In with Google
<table>
<thead>
<tr>
<th>OAuth2 Authorization Code</th>
</tr>
</thead>
<tbody>
<tr>
<td># request parameter, seen server side</td>
</tr>
<tr>
<td>2cohef1vnydr1f9ph1z6dssle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OAuth2 Access Token</th>
</tr>
</thead>
<tbody>
<tr>
<td># populated server side</td>
</tr>
<tr>
<td>93io4x7dxiowf2tmisctvwevzj</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OAuth2 Refresh Token</th>
</tr>
</thead>
<tbody>
<tr>
<td># populated server side</td>
</tr>
<tr>
<td>dyr2pcoq06n5l6kzfn76v11z6</td>
</tr>
</tbody>
</table>
{  
  "email": "cal@fbsdata.com",
  "phone_number": null,
  "sub": "20150522131651406093000000",
  "name": "Cal Test Heldenbrand",
  "given_name": "Cal",
  "family_name": "Heldenbrand",
  "middle_name": "Test",
  "address": {
    "formatted": "3415 39th St S, Fargo, ND 58104",
    "street_address": "3415 39th St S",
    "locality": "Fargo",
    "region": "ND",
    "postal_code": "58104"
  },
  "preferred_username": "fgo.cal",
  "website": "http://flexmls.com",
  "MemberMlsId": "20000809145531659995000000",
  "OfficeKey": "20010102182223656548000000",
  "MemberNrdsId": "1234578"
}
[{
  "Id": "20121119215912756672000000",
  "ResourceUri": "/v1/listings/20121119215912756672000000"
  "StandardFields": {
    "ApprovalStatus": true,
    "ArchitecturalStyle": "2 Story",
    "BathsFull": 4,
    "BedsTotal": 6,
    "BuildingAreaTotal": 4412.0,
    "BuyerAgencyCompensation": "Non-Variable",
    "City": "Fargo",
    "ConstructionMaterials": {
      "DryvitStucco": true,
      "Brick": true,
      "Metal": true
    },
    "CountyOrParish": "Cass",
    "CurrentPrice": 1550000.0,
    "IDXParticipant": true,
    "Latitude": 46.813531,
    "ListOfficeId": "20080922165311188855000000",
    "ListOfficeUserType": "Office",
    "ListPrice": 1550000.0,
    "ListingContractDate": "2012-11-18",
    "ListingId": "12-4922",
    "ListingKey": "20121119215912756672000000"}
Authorization Code Flow

1. Auth Request from Web Browser to OIDC Provider
2. ID Token from OIDC Provider to Web Browser
3. Access Token from OIDC Provider to Web Browser
4. Authorization Code from Web Browser to AWS
5. MLS Data (OData) from RETS Web API to AWS
6. MLS Data (OData) from AWS to API Consumer
The Future of OpenID Connect

● Federated identities
  - Federation is more than just SSO
  - Account linking (ex: Google Identity Toolkit)
  - Trust relationships between Providers (ex: Amazon)

● Federation is currently not part of the OpenID Connect standard!
Account Chooser + Google Identity Toolkit

Demo: gitkitmobile.appspot.com
RETS
Identity Toolkit

- Simple front-end code to include widget
- Populates vendors with configured client_ids
- Saves account state with cookie
- Extend to VOW users?

MLS Member Sign In
Choose your MLS provider

- Sign in with connectMLS
- Sign in with flexmls
- Sign in with InnoVia
- Sign in with Matrix
- Sign in with NAVICA MLS
- Sign in with Paragon
- Sign in with Rapattoni MLS
1) **Member ned.flanders declares** bob@gmail.com **as his client in the MLS**

2) Ned logs into VOW website using the RETS Identity toolkit

3) VOW website queries the MLS's UserInfo Endpoint which returns a list of clients

4) **ned.flanders is now “linked” with** bob@gmail.com

5) Bob logs in at the VOW website using Google+ OpenID Connect login

6) VOW website uses Ned's VOW-scoped access token to retrieve API data for Bob
Demo: TestCMAVendor.com

Test CMA Vendor A Federated OpenID Connect Demo

Check the response_types desired

- ID Token (id_token)
- OAuth2 Authorization Code (code)

- OAuth2 Access Token (token)

OpenID Connect Flow

Authorization Code

Sign In with SparkPlatform

Sign In with Google
Two-Step Verification

Enter the verification code generated by the Authenticator app

874013

Verify

Remember this browser for 30 days

Flexmls® Web
Copyright © 2015 FBS.
All rights reserved.
Test CMA Vendor Callback

ID Token

```json
{
    # populated server side:
    "iss": "https://openidp.fbsdata.com",
    "sub": "20101118193417489550000000",
    "aud": "byjk0hp7x3siwb9xpmcraes",
    "exp": 1432825536,
    "iat": 1432739136,
    "nonce": "12u3jx8",
    "at_hash": "G6GIOfTSVdlnwhRupvkTaw",
    "c_hash": "ST0VX9iuC6AFbiquFFoTiJA",
    "amr": [
        "TOTP"
    ]
}
```
<table>
<thead>
<tr>
<th>OAuth2 Authorization Code</th>
</tr>
</thead>
<tbody>
<tr>
<td># request parameter, seen server side</td>
</tr>
<tr>
<td>1b5e3frp2bfqx9ee6flsumur4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OAuth2 Access Token</th>
</tr>
</thead>
<tbody>
<tr>
<td># populated server side</td>
</tr>
<tr>
<td>bbg9vwlax7se5r495shg8bxmx</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OAuth2 Refresh Token</th>
</tr>
</thead>
<tbody>
<tr>
<td># populated server side</td>
</tr>
<tr>
<td>7swxe9wfzdf6y779dzzm58u5b</td>
</tr>
</tbody>
</table>
# populated server side
{
  "email": "cal@fbsdata.com",
  "phone_number": null,
  "sub": "20150522131651406093000000",
  "name": "Cal Test Heldenbrand",
  "given_name": "Cal",
  "family_name": "Heldenbrand",
  "middle_name": "Test",
  "address": {
    "formatted": "3415 39th St S, Fargo, ND 58104",
    "street_address": "3415 39th St S",
    "locality": "Fargo",
    "region": "ND",
    "postal_code": "58104"
  },
  "preferred_username": "fgo.cal",
  "website": "http://flexmls.com",
  "MemberMlsId": "20000809145531659995000000",
  "OfficeKey": "20010102182223656548000000",
  "MemberNrdsId": "1234578"
}
Amazon DynamoDB Data

```json
# populated server side
[
    {
        "StreetName": "7th",
        "ListingId": "1234",
        "StreetNumber": "1234",
        "StreetDirPrefix": "E",
        "StreetSuffix": "St",
        "ListPrice": "200000.0"
    },
    {
        "StreetName": "8th",
        "ListingId": "5678",
        "StreetNumber": "900",
        "StreetDirPrefix": "W",
        "StreetSuffix": "St",
        "ListPrice": "150000.0"
    }
]
```
Properties From Spark API

```
# populated server side

[
  {
    "Id": "2012119215912756672000000",
    "ResourceUri": "/v1/listings/2012119215912756672000000",
    "StandardFields": {
      "ApprovalStatus": true,
      "ArchitecturalStyle": "2 Story",
      "BathsFull": 4,
      "BedsTotal": 6,
      "BuildingAreaTotal": 4412.0,
      "BuyerAgencyCompensation": "Non-Variable",
      "City": "Fargo",
      "ConstructionMaterials": {
        "DryvitStucco": true,
        "Brick": true,
        "Metal": true
      },
      "CountyOrParish": "Cass",
      "CurrentPrice": 1550000.0,
      "IDXParticipant": true,
    }
  }
]```
Federated Authorization Code Flow

1. Auth Request
2. Authorization Code
3. ID Token
4. Access Token
5. Refresh Token
6. MLS Data (OData)
7. MLS B
The Future of OpenID Connect

- **Proof of Possession access tokens**
  - (AKA Holder of Key token)
  - Uses public/private keys and JWT
  - Cannot be copied and replayed

- **OAuth2 signed requests**
  - Uses PoP tokens to sign each API request
  - TLS is optional!

- **Token Binding**
  - ID and access tokens are “bound” to a TLS session
  - Cannot be replayed
  - Federation – crypto-bind tokens to multiple TLS sessions between many relying parties & providers
Conclusion

- Worldwide standard
- OIDC is a [simple] addition on top of OAuth2
  - Easy migration from Web API Security 1.0.2
- Discovery service removes requirement for a RESO Web Security document
- Certification program will be avail soon
- Once the basics are implemented, many cool features are possible in the future
Questions?