Leveraging OpenID
To connect Vehicle to the Cloud
Who Are We?

Commits by Company in 2017

<table>
<thead>
<tr>
<th>Company</th>
<th>Commits</th>
</tr>
</thead>
<tbody>
<tr>
<td>IoT.bzh</td>
<td>1039</td>
</tr>
<tr>
<td>Linux Foundation</td>
<td>79</td>
</tr>
<tr>
<td>Konsulko</td>
<td>77</td>
</tr>
<tr>
<td>Individual</td>
<td>38</td>
</tr>
<tr>
<td>Mentor Graphics</td>
<td>36</td>
</tr>
<tr>
<td>TI</td>
<td>25</td>
</tr>
<tr>
<td>Advanced Telematics Systems</td>
<td>21</td>
</tr>
<tr>
<td>Xevo</td>
<td>20</td>
</tr>
<tr>
<td>AisinAW</td>
<td>7</td>
</tr>
<tr>
<td>ALPS</td>
<td>7</td>
</tr>
<tr>
<td>Mitsubishi Electric</td>
<td>7</td>
</tr>
<tr>
<td>Renesas</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company</th>
<th>Commits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toyota</td>
<td>5</td>
</tr>
<tr>
<td>Fujitsu-Ten</td>
<td>4</td>
</tr>
<tr>
<td>Intel</td>
<td>4</td>
</tr>
<tr>
<td>Samsung</td>
<td>4</td>
</tr>
<tr>
<td>ADIT</td>
<td>3</td>
</tr>
<tr>
<td>Panasonic</td>
<td>3</td>
</tr>
<tr>
<td>Qt Company</td>
<td>3</td>
</tr>
<tr>
<td>Microchip</td>
<td>2</td>
</tr>
<tr>
<td>Trust Point Innovation</td>
<td>2</td>
</tr>
<tr>
<td>LG</td>
<td>1</td>
</tr>
</tbody>
</table>

1394 Total Commits
45 Committers
21 Companies

- 01 Jan 2017 – 26 May 2017
- Commits to master
V2C Multiple Requirements

- Car to Cloud
  - Telematics
  - Car sharing, Fleet management
  - Profiling
  - Real time Update Traffic/Map
- Cloud to Car
  - User Preferences
  - SOTA, Streaming Music, Traffic
- Car to Infrastructure
  - Payment
  - Car to City
  - Car to Home
V2C MUST fix issues

- Potential open door for cyber-attack?
- Who own and controls the data? What’s about user privacy?
- How to provide the right user experience with on time to the market innovations?
- How to open popular to non-automotive services (Spotify, Facebook, Paypal, …)
- How to keep the service running for 25 years?
- …
- Last but not least, where to find skill developers?
AGL Microservices Architecture

Cluster
- Head Unix
  - Direction Indication
- Transport & ACL
  - Cluster Virtual Signal
    - Engine-CAN-BUS
    - ABS

Entertainment
- Navigation Service
  - Carte handling
  - Localisation management
  - POI
- CAN-BUS Virtual Signal
  - CAN-BUS
  - LIN-BUS
- Geopositioning Virtual Signal
  - Gyro, Accelerometer
  - CAN GPS

Cloud
- My Car Portal
  - Payment
  - Subscriptions
  - Preference
- Maintenance Portal
  - Know Bugs
  - Maintenances
  - Service Packs
- Log Analytics
  - MongoDB Engine
  - Payment Service
  - No-SQL Engine
  - Statistics & Analytics

Multi ECU & Cloud Aware Architecture
AGL-DD API Description Model

static struct afb_auth auths[] = {
    { .type = afb_auth_Permission, .text = "urn:AGL:permission:low-can:partner:read" },
    { .type = afb_auth_Permission, .text = "urn:AGL:permission:low-can:partner:write" },
    { .type = afb_auth_And, .first = &auths[0], .next = &auths[1] }
};

static const struct afb verb v2 verbs v2[] = {
    {.verb = "subscribe", .callback = f_subscribe, .auth = &auths[2], .session = AFB_SESSION_CHECK, },
    {.verb = "unsubscribe", .callback = f_unsubscribe, .auth = &auths[2], .session = AFB_SESSION_CHECK,},
    {.verb = NULL }
};

/* the integer data used by binder for the verbosity of the binder */
int afbBindingV2verbosity;

/* the structure for describing the binder */
const struct afb_binding_v2 afbBindingV2 = {
    .api = "low-can", .specification = /* the JSON description */ .verbs = verbs,
    .init = NULL , .start = start_low_can, .onevent = NULL,
};
OpenAPI Binding Description

```
"openapi": "3.0.0",
"$schema": "file:schema-agl-api-v2.json",
"info": {
  "description": "Can Signal Low Level API", "title": "low-can", "version": "2.0"
},
"servers": [{
  "url": "ws://{host}:{port}/api/low-can", "description": "The API server.",
  "variables": { "host": { "default": "localhost" }, "port": {"default": "1234"}
  },
  "X-afb-events": [ 
    { "$ref": "#/components/schemas/afb-event" }
  ]
},
"components": {
  "schemas": {
    "afb-reply": {
      "properties": {
        "jtype": {
          "type": "string"
        },
        "request": {
          "$ref": "#/components/schemas/afb-request"
        },
        "response": {
          "type": "object"
        }
      }
    }
  }
}
```
AGL-DD Security Model

Not ready yet for Cloud SaaS
Why OpenID Connect?

- Inherit from SAML2 protocols models
  - Over 10 years of lesson learn on massive deployment
  - Support of privacy and data protection built in
- Simpler to deploy than SAML2
  - Low level based on REST, SSL, JSON
  - High Level based on oAuth2, JWT (Json Web Token), JWS (Json Web Signature)
  - Toolkit available in multiple languages
  - Supported natively or flavoured by many internet providers (Facebook, Google, Paypal, …), but also by many governments
- Community
  - Active & well known
  - Open to custom profile
  - Ready to work with AGL
OpenID members

Companies involve OpenID Development
Contributors included a diverse international representation of industry, academia and independent technology leaders: AOL, Deutsche Telekom, Facebook, Google, Microsoft, Mitre Corporation, mixi, Nomura Research Institute, Orange, PayPal, Ping Identity, Salesforce, Yahoo! Japan, among other individuals and organizations.
OpenID Simple Flow

Slide Credit: Nov Matake, OpenID Japan
OpenID Connect Detail Flow

1. Starting at your domain login page, the user clicks on a sign-in with `<Provider>` link.
   - User Agent (Browser): GET/authorize
   - Relying Party (Client App): 302 Redirect to provider authz endpoint
   - Authorization Server: 302 Redirect to provider login

2. The browser renders the login page, and the user authenticates.
   - User Agent (Browser): POST credentials/<Provider>/login/endpoint
   - Relying Party (Client App): 200 Login page
   - Authorization Server: 200 Consent page

3. The browser renders the consent page, and the user authorizes.
   - User Agent (Browser): POST consent/<Provider>/authz/endpoint
   - Relying Party (Client App): 302 Redirect to RP Callback
   - Authorization Server: 200 Respond with tokens

4. The browser renders the home page.
   - User Agent (Browser): GET/client/callback
   - Authorization Server: POST/<Provider>/token/endpoint
   - Relying Party (Client App): 200 Respond with just claim

1. The authorization server checks for an existing session. If no session exists, it redirects the user to log in.

2. If the scopes in the request are not already authorized, the server presents the user with an authorization page.

3. On successful authorization, the server redirects the user to the client app's redirect URL with an authorization code.

4. The Authorization Code is exchanged for an ID Token and Access Token. The ID Token is a signed JWT containing basic claims. The Access Token can be used to access further User Claims at the `user_info` endpoint.

5. The RP verifies the ID Token. If valid, the user is considered authenticated and a session is established.

6. The RP optionally uses the access token to request additional claims from the `user_info` endpoint.
Global Architecture

Local Binding

ws-client:tcp://hostname:port/MyAPI

(1) Request API

(2) Request AuthZ (clientID@IDP, scope, ..)

(3) Forward AuthZ Request

(9) Return AuthCode

(10) Forward AuthCode

Identity Agent

(4) Request AuthZ on behalf Remote (clientID, scope, ..)

(5) Redirect Authentication URL for User consent

(6) User Consent/Authentication Interaction

(7) Forward IDP redirect

(7) User Consent/Authentication Interaction

Consent/Authentication

User UI

Remote Binding

ws-server:tcp://hostname:port/MyAPI

(11) Forward AuthCode

(12) Provide AuthCode

(13) Receive User Info

IDP (Identity Provider)

e.g. www.mycarportal.net
Data Model

Identity Agent Data Structure

(UsrID) Local User Profile
- Name
- Email
- Etc.

(AppID) Local App Profile
- ClientID
- Authority, Scope
- Session Token/Timeout
- Persistant Data
- Etc.

(FedID) IDP pseudonym
- usrID@IDP
- usrProfile@IDP
- AuthZ token
- Session Token/Timeout
- Etc.
Work To Be Done

- AGL Binding Protocol Extension
  - Native integration of OpenID Connect
  - Support for use interaction (consent, authentication)

- Access Controls
  - LOA
  - Hook for roles/group
  - Link with existing privilege model

- Authentication
  - Webview for Authentication/Consent
  - Map authentication devices (NFC, FiDO)
  - Define API for custom API
Further Information

- Specifications: [http://openid.net/connect](http://openid.net/connect)
- Introduction [http://openid.net/connect/faq](http://openid.net/connect/faq)

- Deep dive in protocols: [Following videos are pretty technical, while they relates to one of previous live project they may help to understand OpenID protocols. Please ignore 1st videos which are related to the installation of the project, last ones demonstrate protocols through a live debug session]
  - French [http://breizhme.net/fr/video-technique](http://breizhme.net/fr/video-technique) (2nd & 3rd videos)
  - English [http://breizhme.net/en/](http://breizhme.net/en/) (last video)

**Warning**: When searching for information you should be aware that OpenID-connect has 100% different from OpenID-v1/v2.