Protecting Web Services in Practise: ZXID API

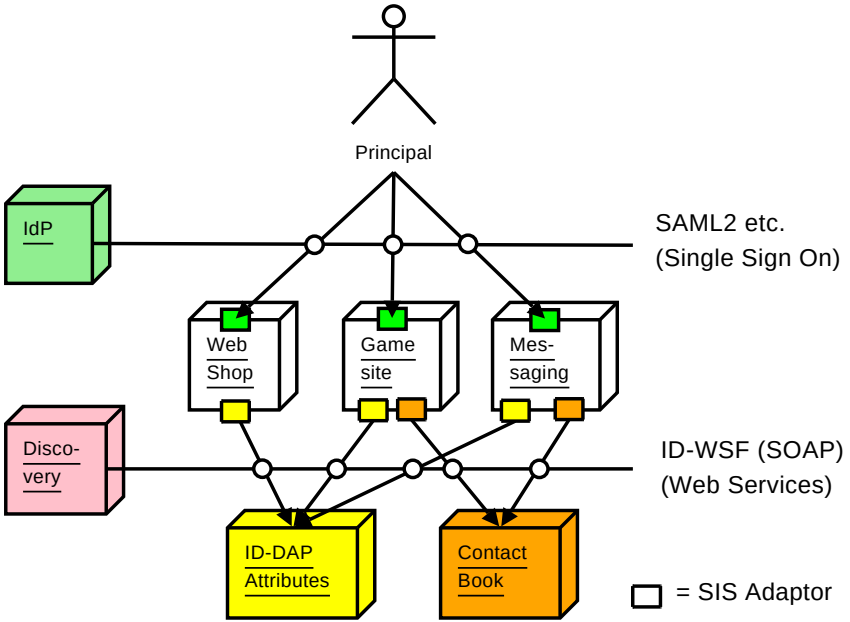
Liberty Alliance SIG-Norway Workshop

Oslo, Aug 28, 2008

Sampo Kellomäki (sampo@symlabs.com)

1. Introduction

- Specs in two layers:
  - SSO (front channel, browser) and
  - ID-WSF (back channel, RPC)
- Connectors and APIs needed to integrate to *your* app
- Different platforms need different solutions
  - Apache front channel (mod_auth_saml, mod_mellon, etc.)
  - Web server independent front channel: do it in your app - platform dependent API
  - Back channel tends to be dependent on programming language and platform of the app


Principal
IdP
SAML2 etc.
(Single Sign On)
Web Shop
Game site
Mes-saging
Disco-very
ID-WSF (SOAP)
(Web Services)
ID-DAP Attributes
Contact Book
= SIS Adaptor

2. Examples of ID-SIS standards

- CB - Contact Book
- CSM - Content Services Messaging (aka ID-Messaging)
- ID-DAP - Identity based Directory Access Protocol
  - with extensible data schema
- PP - Personal Profile
- EP - Employee Profile
- GeoLoc

3. Examples of ID-SIS like specs outside Liberty

- ID-HR-XML
  - about to be brought into Liberty Services Group
- Medical drafts
  - ID-SIS Personal Health Record (PHR)
  - ID-SIS Continuity of Care Record (CCR)
- Payment service (Fidelity research project; eBiz Mobility)
- Calendar (Fidelity research project)
- **Your own service?**

4. ZXID Free and Open

- Open source, IPR clean: Apache 2 License for ZXID itself
  - OpenSSL under BSDish (with "advertising" clause)
  - libcurl under BSDish
  - zlib under BSDish
  - libc available as part of operating system
  - No other dependencies or IPR worries
- Based on open specifications with Royalty Free (RF) licensing
- Free as in freedom *and* free as in free beer
- Sampo's pet project, sponsored by Symlabs: my vision on how things should be done
- Contribution invited
- No collaboration site yet, just mail the author
- Commercial support available from (at least) Symlabs

5. ZXID Easy

- Simple API that hides complexity of SAML and ID-WSF protocols
- No need to understand protocol or even detailed flows
- Simple API covers the 90% cases, low level API gives full power
- Very similar procedural API/binding across all languages
  - C / C++
  - Perl (Net::SAML module)
  - PHP
  - Java
  - C# (csharp) (alpha)
  - More to come as I crank SWIG, e.g. Ruby and Python

5. ZXID Easy 2

- Ready made integration to popular applications
  - mod_auth_saml for Apache httpd
  - DokuWiki
  - More to come as I have time - consider contributing
- Complete stack: one solution covers SAML and ID-WSF
- Over 150 pages of documentation (mostly written by a human)

6. ZXID Easy to Deploy

- AutoCoT: Build Circle of Trust on-the-fly using SAML Well Known Location (WKL) method of metadata exchange
- IdP URL: IdP discovery similar to "light" competition
- None to few (OpenSSL, zlib, libcurl) other modules to install
- All generated code shipped as `.h` and `.c` files - no need to install generation tools or understand schemas.

7. ZXID: Light Weight SAML

- Light weight means
  - Easy to learn: canned tutorial
  - Easy to integrate to existing web application
  - Easy to configure: few and easy to understand configuration items
  - Easy to deploy: minimal external dependencies
  - Efficient implementation (core written in C)
- SAML 2.0 can be deployed in light weight manner
  - Turn off signing (and even TLS if you do not need it)
  - Dynamic trust model (e.g. Auto-CoT and IdP URL)
  - Easy toolkit: ZXID ships with lightweight SAML configured by default
  - `zxid.org` deployed this way is easier and ligher than OpenID
- ID-WSF and SOAP complexity hidden
  - Automatic discovery
  - Programmer concentrates to application layer (SOAP Body)

8. ZXID Focus

- Ready now
  - SAML 2.0 SP
  - ID-WSF 2.0 WSC and WSP
- Soon
  - SAML 1.x
  - XACML
- Eventually
  - ID-FF 1.2, ID-WSF 1.1
  - WS-Federation, WS-Trust
  - Server roles
- ZXID Participates in OpenLiberty as OpenLiberty-ZXID
  - Interop with other OpenLiberty modules
  - API harmonization

9. Acronym Expansion

ID-WSF Liberty Alliance Identity Web Services Framework

IdP Identity Provider (SAML role, asserting party, credential authority)

SP Service Provider (e.g. web site) (SAML role, relying party)

EPR End Point Reference (URL + metadata and possibly credentials)

WSC Web Services Client

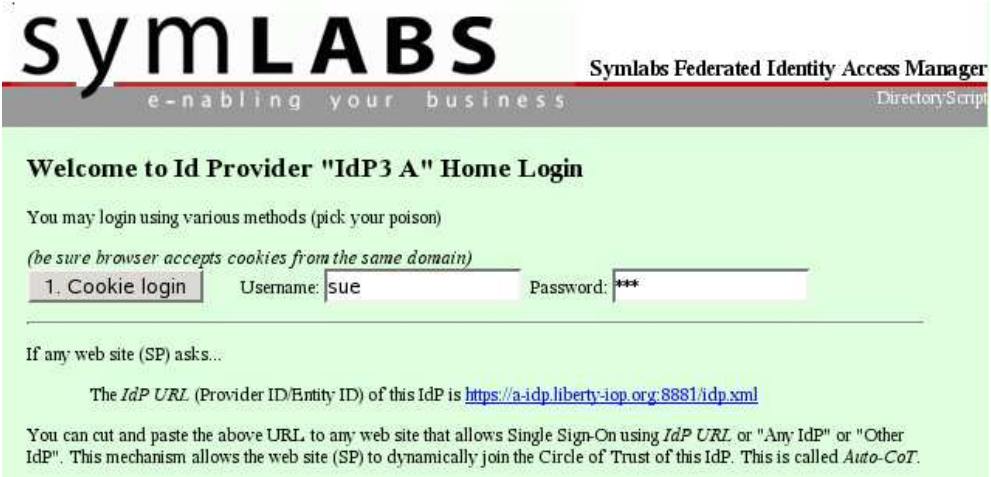
WSP Web Services Provider

DS Discovery Service

DST Data Services Template

IdP Selection

Login at IdP


symLABS
e-nabling your business
Symlabs Federated Identity Access Manager
DirectoryScript
Welcome to Id Provider "IdP3 A" Home Login
You may login using various methods (pick your poison)
(be sure browser accepts cookies from the same domain)
1. Cookie login
Username: sue
Password: ***
If any web site (SP) asks...
The IdP URL (Provider ID/Entity ID) of this IdP is https://a-idp.liberty-iop.org:8881/idp.xml
You can cut and paste the above URL to any web site that allows Single Sign-On using IdP URL or "Any IdP" or "Other IdP". This mechanism allows the web site (SP) to dynamically join the Circle of Trust of this IdP. This is called Auto-CoT.

SSO Successful: Protected Page

SAML Hello World in PHP

- 38 lines of PHP code of which only 22 do something (rest are comments or HTML)
- Complete
  - All profiles are handled
  - Single Logout handled
  - Well Known Location (WKL) metadata exchange handled
- Hides SAML protocol details
- This Hello World can be cut-and-pasted into any PHP application

Initialization once

```
01 <?
02 dl("php_zxid.so");  # Pull in module (.so file)
03 # CONFIG: You must have created /var/zxid directory hierar
04 # CONFIG: You must edit the URL to match your domain name
05 $conf = "PATH=/var/zxid/
            \&URL=https://sp1.zxidsp.org:8443/zxidhlo.php";
06 $cf = zxid_new_conf_to_cf($conf);
07 ?>
```

- PATH configuration means multiple instances of ZXID can coexist (e.g. virtual hosting of web sites)
- URL configuration determines provider ID, can also be configured via `/var/zxid/zxid.conf`

Per protected page or until session is bootstrapped

```
08 <?
09 $qs = $_SERVER['REQUEST_METHOD'] == 'GET'
10     ? $_SERVER['QUERY_STRING']
11     : file_get_contents('php://input');
12 $res = zxid_simple_cf($cf, -1, $qs, \&ses, 0x1814);
13
14 switch (substr($res, 0, 1)) {
15 case 'L': header($res); exit;
16 case '<': header('Content-type: text/xml'); echo $res; exi
```

- Read input and call *zxid_simple()* to handle SAML protocol details
- Act on outcome of *zxid_simple()* as indicated by the first letter
  - L: protocol requires redirect, perform it
  - <: Send out XML data (such as Metadata or SOAP response)

The IdP Selection Page

```
17 case 'n': exit;   # Already handled, do nothing further
18 case 'e':
19 ?>
20 <title>Please Login Using IdP</title>
21 <h1>Please Login Using IdP</h1>
22 <?=zxid_idp_select_cf($cf, null, 0x1800)?>
23 <?
24 exit;
```

- e: indicates that IdP Selection page needs to be rendered
- *zxid_idp_select()* generates the ZXID standard form
- Altenatively you could supply your own HTML for the form as long as you respect the form field naming convention

Login Successful Case

```
25 case 'd': break;  # Logged in case -- continue after switc
26 default:  die("Unknown zxid_simple() res($res)");
27 }
28
29 # Parse the LDIF in $res into a hash of attributes $attr
30
31 foreach (split("\n", $res) as $line) {
32     $a = split(": ", $line);
33     $attr[$a[0]] = $a[1];
34 }
35 ?>
```

- d: login successful, return data is LDIF entry with attributes of SSO

Protected Content with Single Logout and Defederate Buttons

```
36 <title>Protected content, logged in</title>
37 <h1>Protected content, logged in as <?=$attr['cn']?></h1>
38 <?=zxid_fed_mgmt_cf($cf, null, -1, $attr['sesid'], 0x1800)
```

- *zxid_fed_mgmt()* generates the Single Log-Out buttons
- This is the place to bootstrap your application's own session

Login Successful: Returned LDIF

```
dn: idpnid=Pa45XAs2332SDS2asFs,affid=https://idp.demo.com/
objectclass: zxidsession
affid: https://idp.demo.com/idp.xml
idpnid: Pa45XAs2332SDS2asFs
authnctxlevel: password
sesid: S12aF3Xi4A
cn: Joe Doe
```

- The LDIF entry is used as convenient format for passing attribute-value pairs from *zxid_simple()* to application
- Some "attributes" are synthesized, others come actually from assertion

10. Liberty ID-WSF (Web Services Framework)

- Focuses on passing identity on SOAP calls (just what SOA needs)
- Secures the SOAP call
  - WS-Security header
  - SAML token conveying authenticated user present and consented
  - Digital signature
- Helps locate SOAP services
  - Bootstrap
  - Discovery
- Calling other user's web services: People Service
- User consent querying: Interaction Service

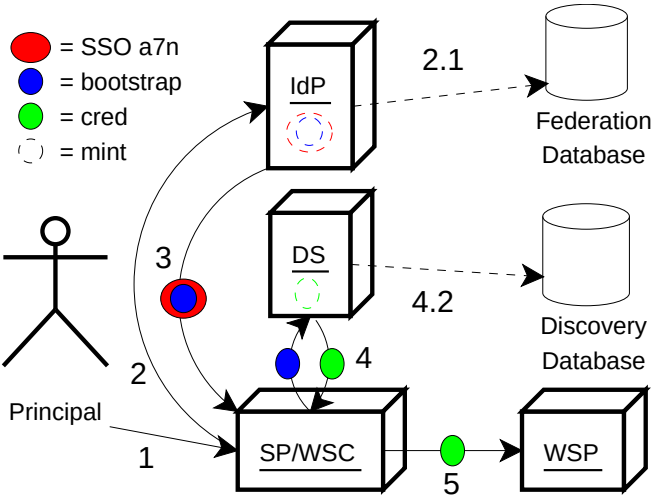
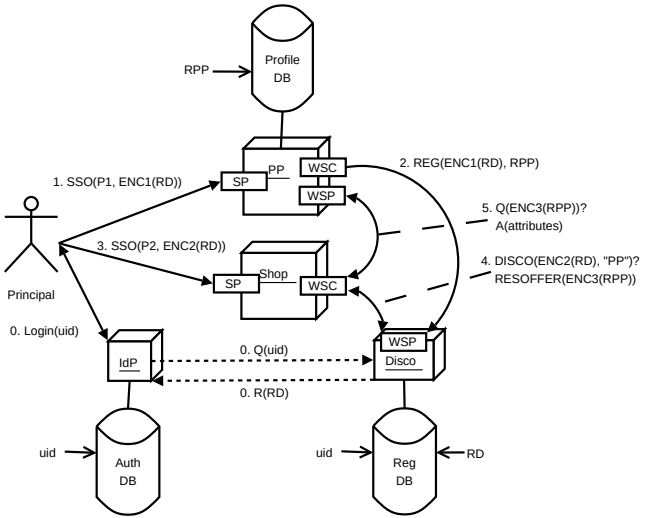


Figure 1: Single Sign-On (2,3), Discovery (4), and call to WSP (5). The blue ball represents discovery bootstrap.


RPP
Profile
DB
PP
SP
WSC
WSP
1. SSO(P1, ENC1(RD))
2. REG(ENC1(RD), RPP)
5. Q(ENC3(RPP))?
A(attributes)
3. SSO(P2, ENC2(RD))
4. DISCO(ENC2(RD), "PP")?
RESOFFER(ENC3(RPP))
Principal
Shop
SP
WSC
0. Login(uid)
IdP
0. Q(uid)
WSP
Disco
0. R(RD)
uid
Auth
DB
uid
Reg
DB
RD

ID-WSF: Simplest Flow

1. SSO (e.g. with SAML 2.0), gives bootstrap
   (a) Location of web service
   (b) SAML token to be used as credential for web service
2. Call web service with bootrap

ID-WSF: Flow with Discovery

1. SSO (e.g. with SAML 2.0), gives bootstrap
   (a) Location of Discovery service
   (b) SAML token to be used as credential for Discovery service
2. Call Discovery service with bootrap, gives
   (a) Location of web service
   (b) SAML token to be used as credential for web service
3. Call web service

ID-WSF: Discovery registration

1. Register service in general
2. Register association of the service to the user

11. ZXID HR-XML Hello World

- 10 lines of code to set up
- 10 lines to do the actual call
- Hides ID-WSF framework layer details
- Assumes bootstrap EPR was already obtained, e.g. via the SAML SSO shown above
- Cut-and-paste this code to your own programs

  Demo of user experience

ZXID WSC: Preparatory step: SSO

```
01 cf = zxid_new_conf_to_cf(CONF);
02
03 res = zxid_simple_cf(cf, cl, qs2, 0, 0x1fff);
04 switch (res[0]) {
05 default:
06   ERR("Unknown zxid_simple() response(%s)", res);
07 case 'd': break; /* Logged in case */
08 }
09 sid = strstr(res, "sesid: ");
10 zxid_get_ses(cf, ses, sid);
```

- We need the SSO session to have bootstrap

ZXID WSC: Calling HR-XML Modify Method

```
11 env = zxid_callf(cf, ses, zx_xmlns_idhrxml,
12         "<idhrxml:Modify>"
13           "<idhrxml:ModifyItem>"
14             "<idhrxml:Select>%s</idhrxml:Select>"
15             "<idhrxml:NewData>%s</idhrxml:NewData>"
16           "</idhrxml:ModifyItem>"
17         "</idhrxml:Modify>", cgi.select, cgi.data);
18 ZXID_CHK_STATUS(env, idhrxml_ModifyResponse,
19                  hrxml_resp = "Modify failed"; break);
20 hrxml_resp = "Modify OK";
```

- Build call as string
- *ZXID_CHK_STATUS()* macro handles routine error checking

12. ZXID WSC: Building Call as C data

- 16 lines of code
- Parse more complex return value

Prepare Request and make SOAP call

Prior to this, SSO has happened.

```
50 struct zx_e_Envelope_s* env;
51 struct zx_a_EndpointReference_s* epr;
52 epr = zxid_get_epr(cf, ses, zx_xmlns_dap, 1);
53 if (epr) {
54   env = zx_NEW_e_Envelope(cf->ctx);
55   env->Header = zx_NEW_e_Header(cf->ctx);
56   env->Body = zx_NEW_e_Body(cf->ctx);
57   env->Body->Query = zxid_mk_dap_query(cf, ...);
58   env = zxid_wsc_call(cf, ses, epr, env);
```

Process the response

```
59   if (env->dap_QueryResponse) {
60     if (memcmp(env->Body->GetObjectResponse->Status->code-
61       DIE("Bad status");
62     D("Result is LDIF(%.*s)",
63        env->Body->dap_QueryResponse->Data->LDIF->gg.conten
64        env->Body->dap_QueryResponse->Data->LDIF->gg.conten
65   }
66 }
```

13. mod_auth_saml

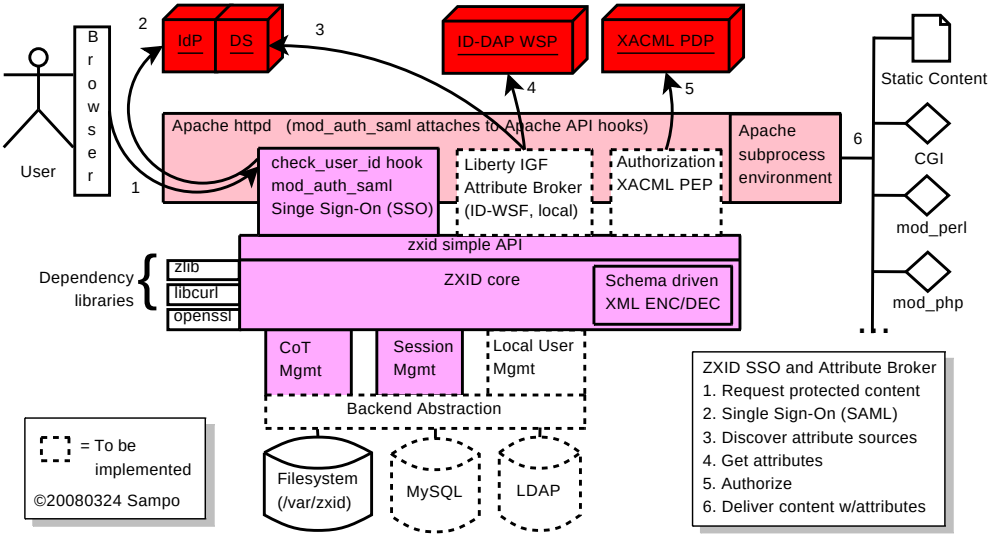


Figure 2: ZXID, via mod_auth_saml, adds to Apache httpd Single Sign-On (SSO), Attribute Broker, and XACML PEP Capabilities that can be used by existing static and dynamic content without alteration

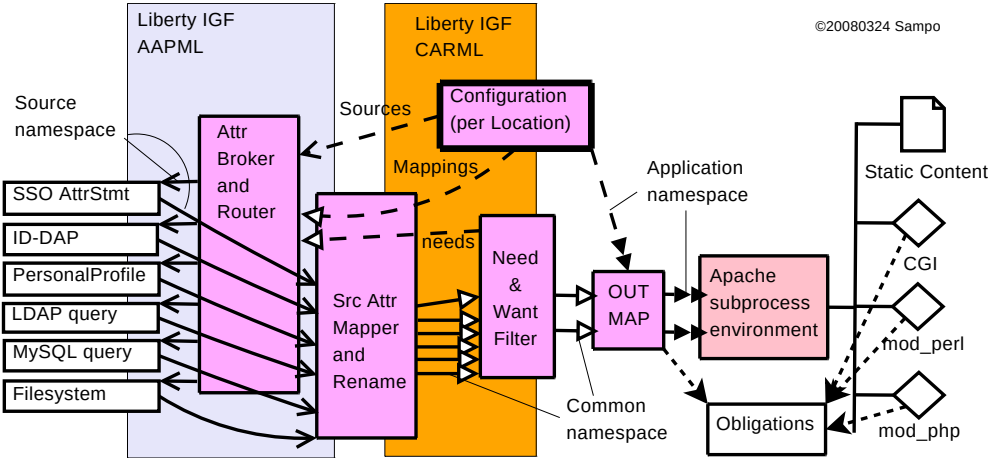


Figure 3: Liberty IGF Compliant Attribute Broker fetches the needed attributes from available providers and populates them to the subprocess environment.

14. SAML Hello World in Java

- Use Java Native Interface (JNI) to expose C library as Java interface
- Java Servlet
- Run under Tomcat
- 53 lines of code

Java: Initialization and Configuration

```
01 import zxidjava.*;
02 import java.io.*;
03 import javax.servlet.*;
04 import javax.servlet.http.*;
05 public class zxidhlo extends HttpServlet {
06   static { System.loadLibrary("zxidjni"); }
07   static final String conf
08     = "PATH=/var/zxid/
          \&URL=http://sp1.zxidsp.org:8080/zxidservlet/zxidHL
```

Call zxidjni.simple(), deal with redirect

```
09   public void do_zxid(HttpServletRequest req,
                         HttpServletResponse res, String qs)
10                       throws ServletException, IOException
11     String ret = zxidjni.simple(conf, qs, 0xd54);
12     switch (ret.charAt(0)) {
13     case 'L':  /* Redirect: ret == "LOCATION: urlCRLF2" */
14       res.sendRedirect(ret.substring(10, ret.length() - 4)
15       return;
```

Deal with verbatim output

```
16      case '<':
17        switch (ret.charAt(1)) {
18        case 's':  /* <se:  SOAP envelope */
19        case 'm':  /* <m20: metadata */
20          res.setContentType("text/xml");
21          break;
22        default:
23          res.setContentType("text/html");
24        break;
25        }
26        res.setContentLength(ret.length());
27        res.getOutputStream().print(ret);
28        break;
```

Logged in: Protected Page and Mgmt Buttons

```
29      case 'd': /* Logged in case */
30        //my_parse_ldif(res);
31        res.setContentType("text/html");
32        res.getOutputStream().print(zxidjni.fed_mgmt(conf, 0
33        break;
34      default:
35        System.err.print("Unknown zxid_simple() response:");
36        System.err.print(ret);
37      }
38    }
```

Map HTTP Methods to ZXID

```
39   public void doGet(HttpServletRequest req, HttpServletRes
40                     throws ServletException, IOException {
41     // LECP/ECP PAOS header checks
42     do_zxid(req, res, req.getQueryString());
43   }
44   public void doPost(HttpServletRequest req, HttpServletRe
45                      throws ServletException, IOException
46     String qs;
47     int len = req.getContentLength();
48     byte[] b = new byte[len];
49     int got = req.getInputStream().read(b, 0, len);
50     qs = new String(b, 0, got);
51     do_zxid(req, res, qs);
52   }
53 }
```

15. Doing it on your own?

- Simplified SOAP Binding
  - Cut features to bare minimum to make it simple (but less generic)
- You still need to pick your poison
  - What development environment / platform and it's idiosyncracies
  - Roll your own or leverage existing modules
    - Learn modules
    - Debug them
    - Integrate them
    - Modules drive the structure and archotecture of your app
- Is this really easier than just learning the full stack API once?

16. Available Toolkits and APIs

- `openliberty.org` is good resource
- Front channel
  - ZXID.org: *zxid_simple()*
  - Lasso
  - OpenSSO
  - Commercial, etc.
- Back channels tend to be dependent on programming language and platform of the app
  - `ZXID.org` WSC and WSP
  - OpenLiberty-J WSC, Conor WSP
  - Commercial, etc.

17. Q&A

- Your questions, please
- `zxid.org`
- `openliberty.org`
- `http://www.projectliberty.org/liberty/specifications__1`
- `http://www.projectliberty.org/liberty/membership`
- Please feel free to contact me or Liberty Alliance staff
- Sampo Kellomäki (sampo@symlabs.com)
  +351-918.731.007

7. What Liberty provides

- standards (for invocation, accounting, security)
- privacy
- security
- auditability
- extensibility
- interoperability
- discovery
- cross prinicpal permission/sharing (people service)

7.1 What Liberty provides 2

- Vendor community
- Deployer community
  - Deployer community input to process
- Forum for getting standards done
- Interops
- Certification
- IPR framework

8. Liberty Services Process

1. Charter, gather interested parties
2. Marketing Requirements
3. Specification
4. Dissemination
   - Potential interop or certification activities
   - Potential reference implementation or open source
   - Tradeshows and PR
5. Revise and improve

8.1 Unofficial Alternative Process

1. Prototype implementation
2. Specification
3. Marketing Requirements
4. Charter
5. Rubber stamp it

8.2 Starting a New Service Specification

- Minimum 3 members needed to propose new spec
  - The more the marrier
  - Without interested parties that contribute work, specs will not go very far
  - Gauge true interest and commitment
- Ideally group should have at least one of each
  - Deployer organization with
    - domain expertise and
    - business case
  - Vendor/implementor organization
  - Liberty expert
  - Possibly end user representation (if different than deployer)

8.3 Work and Timeline

- Minimum theoretical time from idea to spec is about 6 months
- In practise takes at least a year
- Useful outputs can be published much eariler in draft form
- Typical roles
  - Chair (and sometimes vice-chair)
  - Spec editor (often a Liberty expert such as myself)
  - Domain experts
  - White paper author / Marketing oriented person
  - White paper / marketing contributors and reviewers
  - Technical contributors and reviewers
- Methods of work
  - Face to face meetings (often hosted by deployer)
  - Conf calls

9. Recommendations

- Use the Data Services Template / CRUD abstraction *if* doing data service
- Link user identity to the service being accessed
- Use the existing ID-SIS specifications as template, example, guideline, or best practices
- Start small
- Use only what you need
- Create consensus
- Document

9.1 Upgrading an Existing Spec

- If vertical already has a non identity spec, just upgrade it
- Getting a Liberty expert on board early will help you identify how it glues to Liberty world - what can be reused and what should be newly specified
- Success stories
  - Messaging (MM7)
  - Geoloc (MLP)
  - Contact Book (vCard)
  - ID-DAP (LDAP/LDIF)
  - ID-HR-XML (plain HR-XML) (WIP)

2. ID-SIS and its place in Liberty specs

- Vertical specific implemetations of the Liberty architecture
- Build on the framework
- Concrete applications that actually DO something
- ID-SIS spec can act as a profile indicating how things glue together
- ID-SIS spec can be a Service Oriented Architecture (SOA) for a vertical
- Best practices

Liberty Federation Framework ID-FF SAML 2.0

Enables identity federation and management through features such as identity/account linkage Simplified Sign-On, and simple session management.

Liberty Identity Service Interface Specifications (ID-SIS)

Enables interoperable identity services such as personal identity profile, contact book, presence, and so on

Liberty Web Services Framework (ID-WSF)

Provides the framework for building interoperable identity services, permissions based attribute sharing, identity service description and discovery, and the associated security profiles.

Liberty specifications build on existing standards
(SAML, SOAP, WS-Addressing, WS-Security, XML, etc.)