

Creating SAML2 Identity federation for SSO

Sampo Kellomäki (sampo@symlabs.com), Symlabs

MISC2010 WS11: Identity Workshop, Location: Room 1

Wed 20. January, 2010, London

Agenda SAML2 Identity Federation

1. Intro to federation
2. Singel Sign-On Theory
3. Building Circle of Trust
 - (a) Auto federation
 - (b) zxcot tool
4. SSO Practical
 - (a) mod_auth_saml way
 - (b) SSO servlet way
 - (c) *zxid_simple() / tas3_sso()* way

TAS3 Session Agenda

1. TAS3 Architecture
2. ID-WSF Basics
3. ID-WSF Hands-on
 - (a) WSC
 - (b) WSP

SAML2 SSO Federation

What is an Identity Federation

- Agreement by business partners to work together trusting each other to authenticate users and pass identity data about the users.
- Contractual liability to each other and to users
- A scheme for identifying users, to share identifiers. A federation database holding these identifiers.
- Supporting technical infrastructure, usually standards based
- Applicable to Single Sign-On (SSO) and Web Services

What is Single Sign-On (SSO)

- User needs to supply his credentials (e.g. password) only once in a session spanning several web sites
- Sites trust other sites to authenticate users
- Single Sign-On is often used also for attribute push
- Single Logout
- Convenient for users
- Benefits from outsourcing user management, password resets

Identity Federation Roles for SSO

- Service Provider (SP) or Relying Party (RP)
- Identity Provider (IdP)

Trust Network Models

- IdP hub
- SP hub
- Multiple bilateral trust relationships form trust network
- Consortium model of federation agreement

Building Circle of Trust

- Metadata exchange needed for technical integration
- Well Known Metadata approach
 - EntityID is the URL where to get the Metadata
- Trust
 - Trust list approach
 - Online Certificate Status Protocol approach
 - Central Trust approach

Example PEM Cert

```

-----BEGIN          CERTIFICATE-----          MIIGWTCCBcKgAwIBAgIDA-
JEBMA0GCSqGSIb3DQEBBQUAMIIBEjELMAkGA1UEBhMC      RVMxE-
jAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vsb25hM
VQQKEyBJUFMgQ2VydGlmaWNhdGlubiBBdXRob3JpdHkgcy5sLjEuMC
Z2VuZXJhbEBpcHNjYS5jb20gQy5JLkYuICBCLUI2MjlxMDY5NTEuMCwG
aXBzQ0EgQ0xBU0VBMSBDZXJ0aWZpY2F0aW9uIEF1dGhvcmI0eTEuMC
aXBzQ0EgQ0xBU0VBMSBDZXJ0aWZpY2F0aW9uIEF1dGhvcmI0eTEuMC
DQEJARZRZ2VuZXJhbEBpcHNjYS5jb20wHhcNMDYwNDI2MTUzNjU0W
MTUzNjU0WjCBImELMAkGA1UEBhMCUFQxDzANBgNVBAgTBkxpc2JvY
BxMGTGlzYm9hMRMwEQYDVQQKEwpTeW1sYWJzIFNBMRQwEgYDVQ
aWNlczEYMBYGA1UEAxMPaWRwLnN5bWRIbW8uY29tMSAwHgYJKoZI
ZWxpeEBzeW1sYWJzLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwg
x5ZjAl06CZcSMVtjoaS2sCbrBq/whwWnuVgbD6gAM9EO9qDDEs9eB5r
iFTWuZy9jdxL5wNgr2Zk8NxytyaznQgAddKLCSqPZh7Dd+U3Z5hoGtL

```


MDIvaXBzY2EyMDAyQ0xBU0VBMS5jcmwwPaA7oDmGN2h0dHA6Ly93
c2NhLmNvbS9pcHNjYTIwMDIvaXBzY2EyMDAyQ0xBU0VBMS5jcmwwM
AQEEJjAkMCIGCCsGAQUFBzABhhZodHRwOi8vb2NzcC5pcHNjYS5jb20
Sib3DQEBBQUAA4GBACan4TGRFHayR38xPkMabzww9VmCbm0uwPxxk
jkSenPpwpvomvNfp4G0WJdavid7KnZBbMbnKx1qTMgge/ftBnuqcrn6w
aHftQ+r2gFYiVX4HEa6NU5AgpiQjme0Vh3Hzs228lVllgsFqv6YbdlyTYIU
——END CERTIFICATE——

Example PEM Cert And Private Key As Used by ZXID

```
-----BEGIN CERTIFICATE----- MIIGWTCCBcKgAwIBAgIDA-  
JEBMA0GCSqGSIb3DQEBBQUAMIIBEjELMAkGA1UEBhMC RVMxE-  
jAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vsb25hM  
VQQKEyBJUFMgQ2VydGlmaWNhdGlubiBBdXRob3JpdHkgcy5sLjEuMC  
(snip) SIb3DQEBBQUAA4GBACan4TGRFHayR38xPkMabzww9VmCbm0u  
jkSenPpwpvomvNfp4G0WJdavid7KnZBbMbnKx1qTMgge/ftBnuqcrn6v  
aHftQ+r2gFYiVX4HEa6NU5AgpiQjme0Vh3Hzs228lVllgsFqv6YbdlyTYIU  
-----END CERTIFICATE-----
```

```
-----BEGIN RSA PRIVATE KEY----- MIICXQIBAAKBgQDTq7HHImMCXToJlx  
oMMSz14HmfiUcZjxL3iIVNa5nL2N3EvnA2CvZmTw3HK3JrOdCAB10os.  
8nEpzyUJWXpCs9K+kuuJAKAm0b523XnsJmsipA+ZDdyqrUjKDo6WH3  
/GeJEXxqlwfcj2lZLp/iIRvG7ICjN/rdWoNImF3HVBRS -----END RSA  
PRIVATE KEY-----
```


jAQBgNVBAgTCUJhcmNlbG9uYTESMBAGA1UEBxMJQmFyY2Vsb25hM
 VQQKEyBJUFMgQ2VydGlmaWNhdGlubiBBdXRob3JpdHkgcy5sLjEuMC
 (snip) jkSenPpwpvomvNfp4G0WJdavid7KnZBbMbnKx1qTMgge/ftBnuo
 aHftQ+r2gFYiVX4HEa6NU5AgpiQjme0Vh3Hzs228lVllgsFqv6YbdlyTYIU
 <md:ArtifactResolutionService Binding="urn:oasis:names:tc:SAML:2.0:
 Location="https://idp1.zxidp.org:8443/zxididp?o=S"/> <md:SingleLog
 Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
 Location="https://idp1.zxidp.org:8443/zxididp?o=Q" Re-
 sponseLocation="https://idp1.zxidp.org:8443/zxididp?o=Q"/>
 <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindi
 Location="https://idp1.zxidp.org:8443/zxididp?o=S"/> <md:ManageNa
 Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
 Location="https://idp1.zxidp.org:8443/zxididp?o=Q" Re-
 sponseLocation="https://idp1.zxidp.org:8443/zxididp?o=Q"/>
 <md:ManageNameIDService Binding="urn:oasis:names:tc:SAML:2.0:bi


```

<ds:X509Data> <ds:X509Certificate> MIIGWTCCBcKgAwIBAgIDA-
JEBMA0GCSqGSIb3DQEBBQUAMIIBEjELMAkGA1UEBhMC RVMxE-
jAQBgNVBAgTCUJhcmNIbG9uYTESMBAGA1UEBxMJQmFyY2Vsb25hM
VQQKEyBJUFMgQ2VydGlmaWNhdGlubiBBdXRob3JpdHkgcy5sLjEuMC
(snip) jkSenPpwpvomvNfp4G0WJdavid7KnZBbMbnKx1qTMgge/ftBnuo
aHftQ+r2gFYiVX4HEa6NU5AgpiQjme0Vh3Hzs228lVllgsFqv6YbdlyTYIU
<md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindi
Redirect" Location="https://idp1.zxidp.org:8443/zxididp?o=Q"
ResponseLocation="https://idp1.zxidp.org:8443/zxididp?o=Q"/>
<md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindi
Location="https://idp1.zxidp.org:8443/zxididp?o=S"/> <md:ManageNa
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
Location="https://idp1.zxidp.org:8443/zxididp?o=Q" Re-
sponseLocation="https://idp1.zxidp.org:8443/zxididp?o=Q"/>
<md:ManageNameIDService Binding="urn:oasis:names:tc:SAML:2.0:bi

```

```

Location="https://idp1.zxidp.org:8443/zxididp?o=S"/> <md:NameIDFormat:
format:persistent</> <md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:
format:transient</> <md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:
POST-SimpleSign" Location="https://idp1.zxidp.org:8443/zxididp?o=P"
index="5"/> <md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:
POST" Location="https://idp1.zxidp.org:8443/zxididp?o=P"
index="4"/> <md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:
POST" Location="https://idp1.zxidp.org:8443/zxididp?o=S"
index="3"/> <md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:
POST" Location="https://idp1.zxidp.org:8443/zxididp?o=P"
index="2"/> <md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:
Artifact" Location="https://idp1.zxidp.org:8443/zxididp"
index="1"/></></>

```

SSO Servlet Approach for Tomcat

```
01 import zxidjava.*;    // Pull in the zxidjni.az() API
02 import java.io.*;
03 import javax.servlet.*;
04 import javax.servlet.http.*;
05
06 public class zxidappdemo extends HttpServlet {
07     public void doGet(HttpServletRequest req, HttpSe
08         throws ServletException, IOException
09     {
10         String fullURL = req.getRequestURI();
11         if (req.getQueryString() != null)
12             fullURL += "?" + req.getQueryString();
13         System.err.print("Start ZXID App Demo GET("+ful
14         HttpSession ses = req.getSession(false); // Im
```

```
15     if (ses == null) {
16         res.sendRedirect("sso?o=E&fr=" + fullURL);
17         return;
18     }
19
20     res.setContentType("text/html");
21     res.getOutputStream().print("<title>ZXID Demo A");
22
23     // Render logout buttons (optional)
24
25     res.getOutputStream().print("[<a href=\"sso?gl=");
26
```

SAML Hello World in PHP, the *zxid_simple()* approach

- 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
04 # CONFIG: You must edit the URL to match your domain
05 $conf = "PATH=/var/zxid/
           &URL=https://sp1.zxidsp.org:8443/zxidhlo.ph
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 $re
```

- 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 furt
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
- Alternatively 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
26 default: die("Unknown zxid_simple() res($res)");
27 }
28
29 # Parse the LDIF in $res into a hash of attributes $
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']?>
```

```
38 <?=$zxid_fed_mgmt_cf($cf, null, -1, $attr['sesid'], 0
```

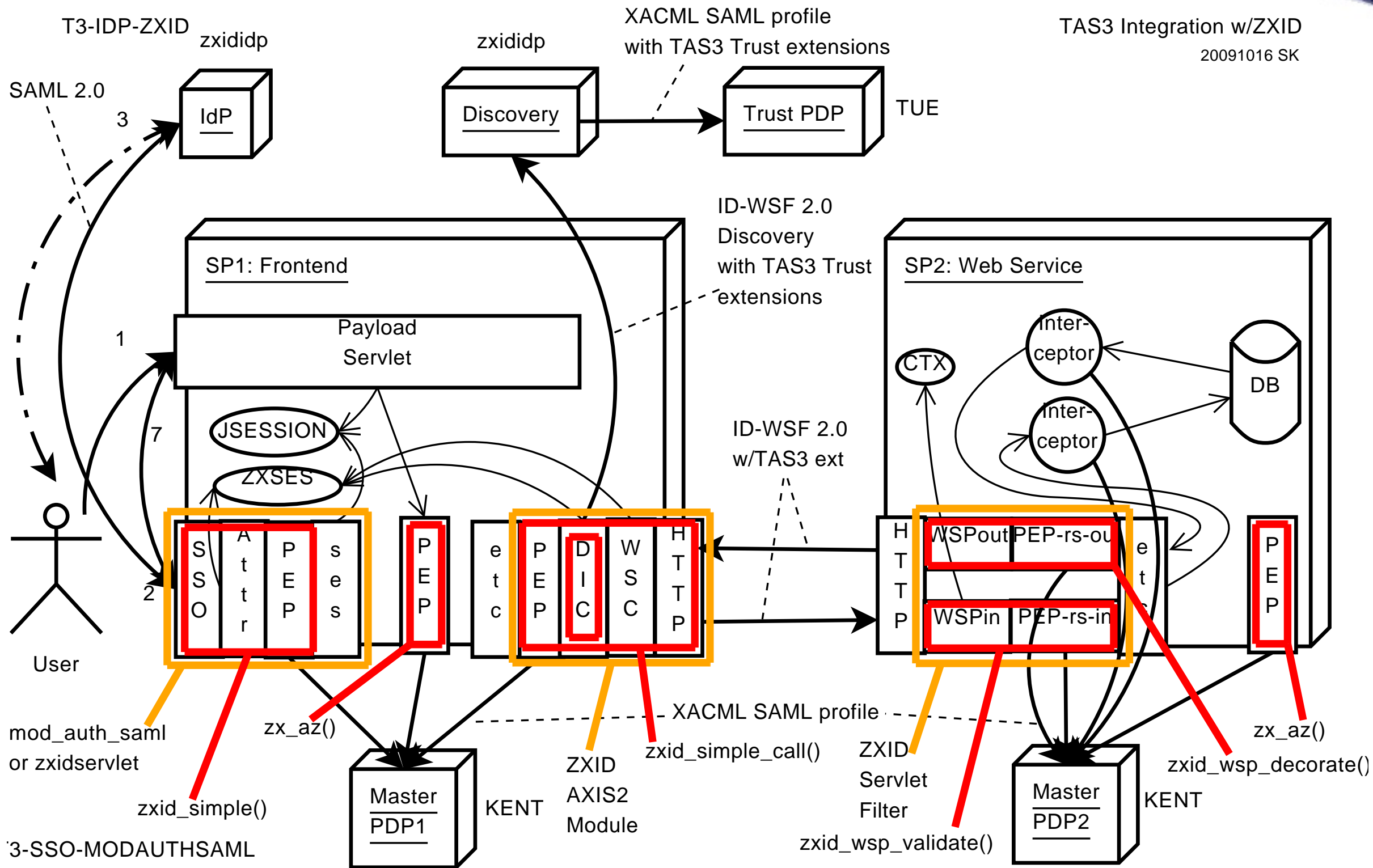
- *zxid_fed_mgmt()* generates the Single Log-Out buttons
- This is the place to bootstrap your application's own session

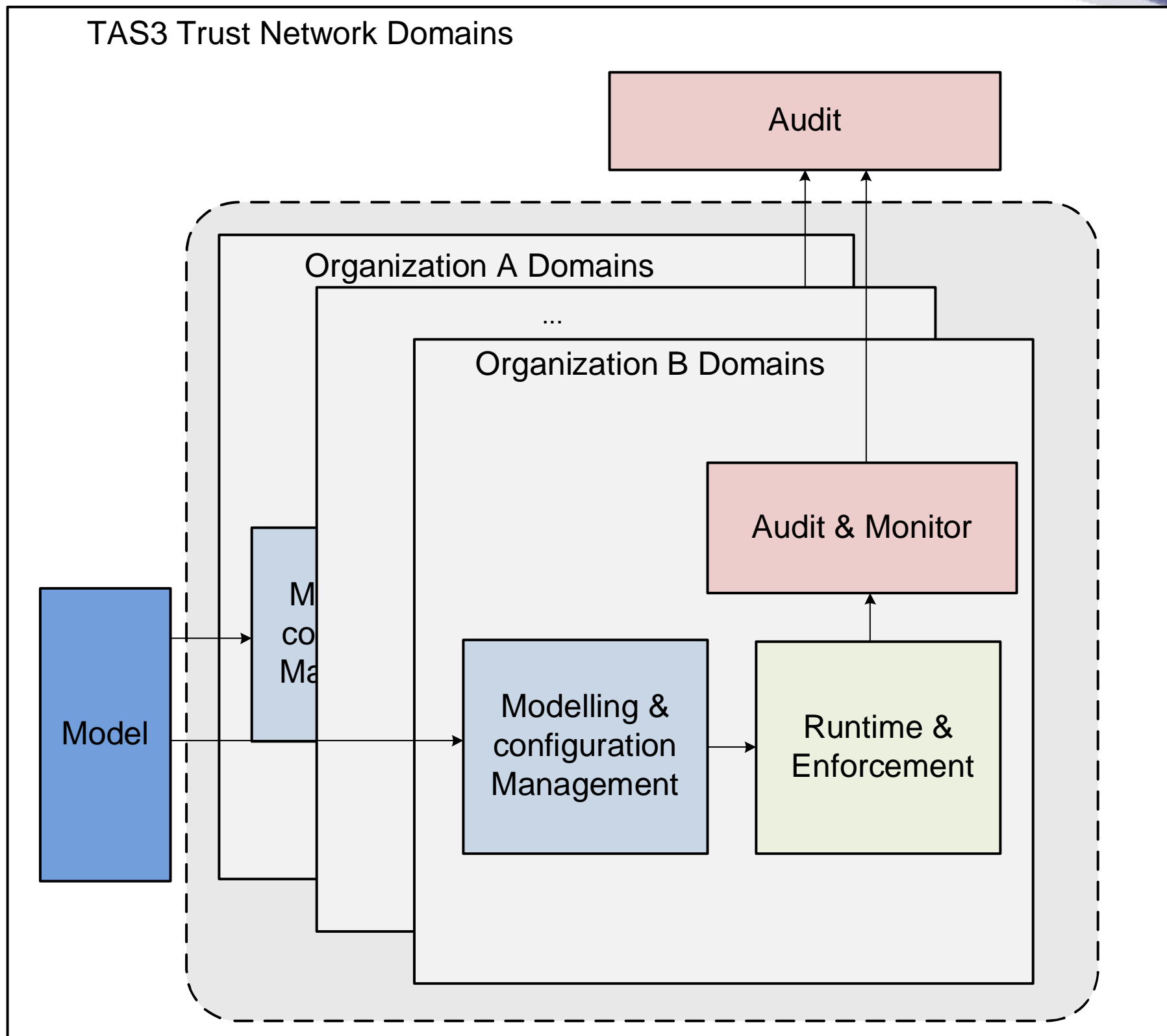
Login Successful: Returned LDIF

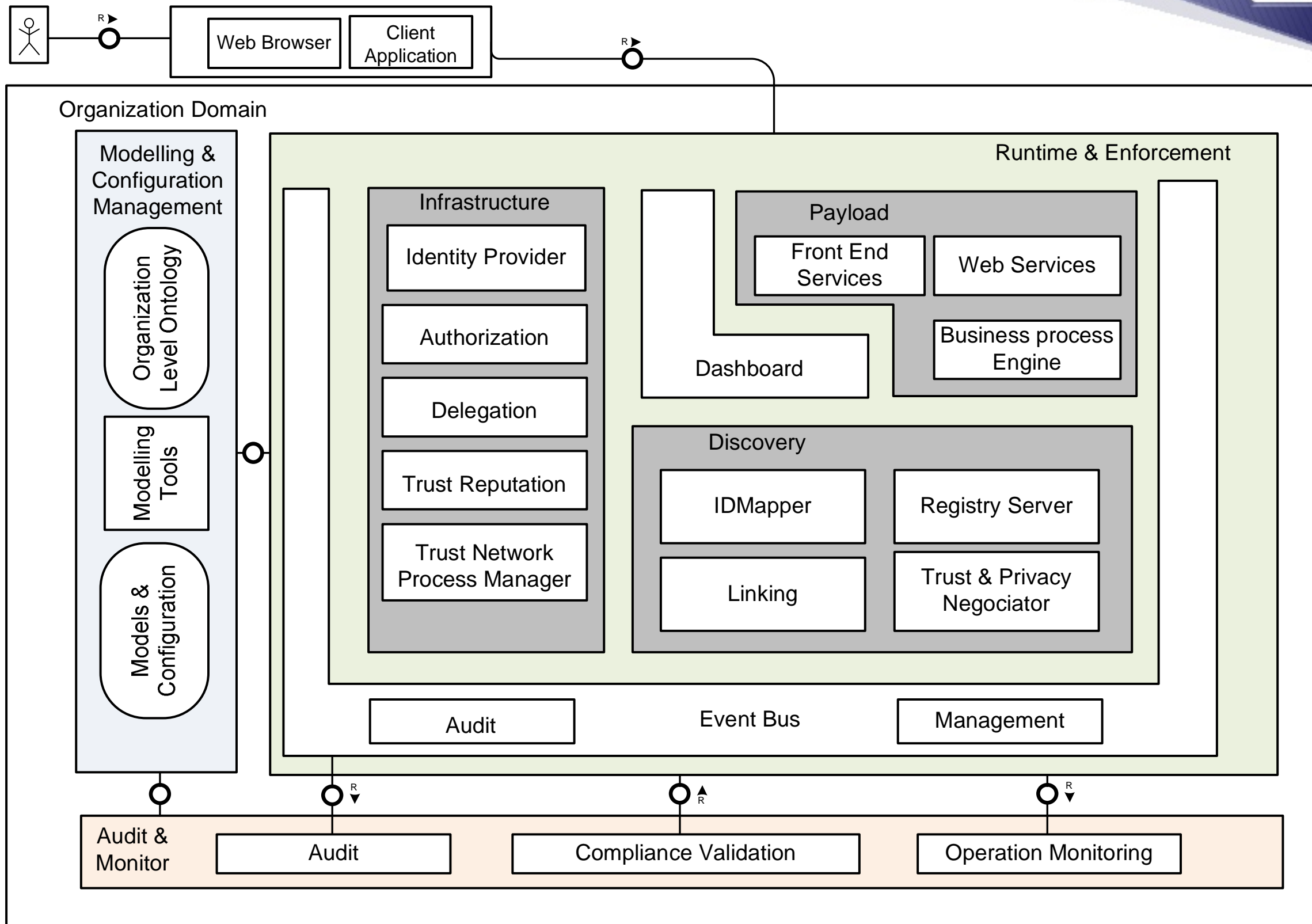
```
dn: idpnid=Pa45XAs2332SDS2asFs,affid=https://idp.dem
objectclass: zxidsession
affid: https://idp.demo.com/idp.xml
idpnid: Pa45XAs2332SDS2asFs
authnctxlevel: password
sesid: S12aF3Xi4A
cn: Joe Doe
```

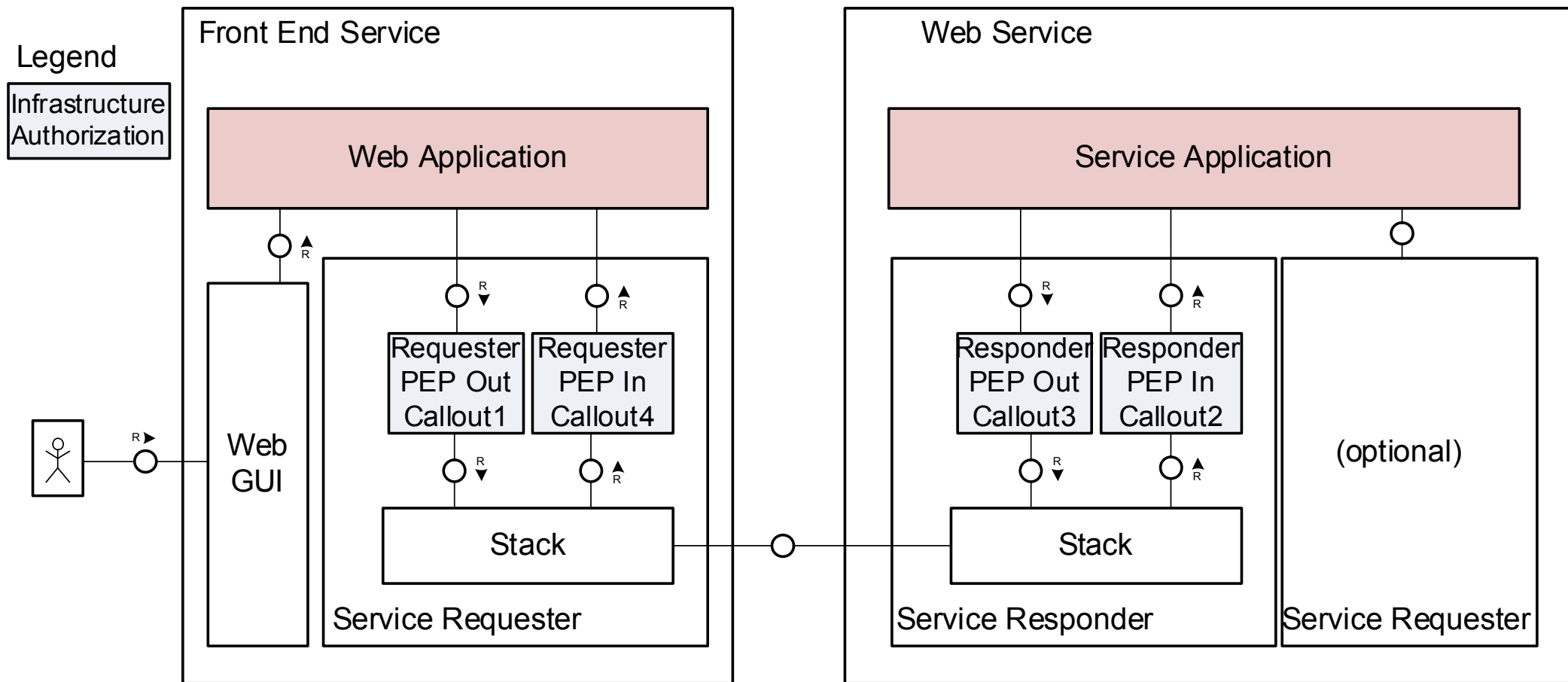
- The LDIF entry is used as convenient format for passing attribute-value pairs from *zxid_simple()* to application
- Some "attributes" are synthesized, others come actually from assertion

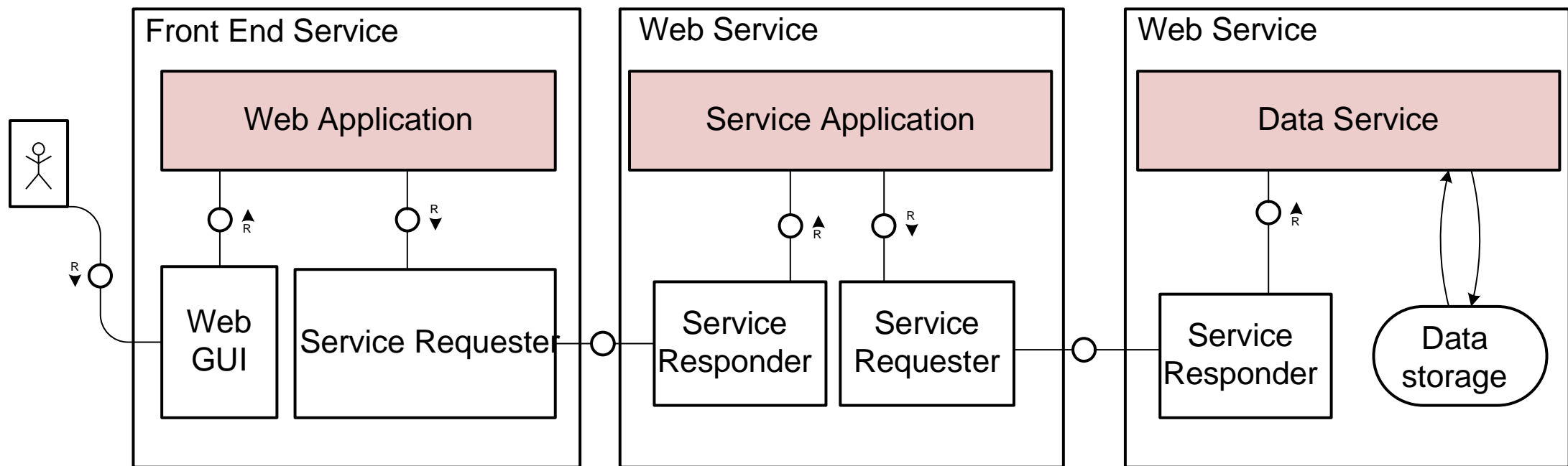
TAS3 Architecture

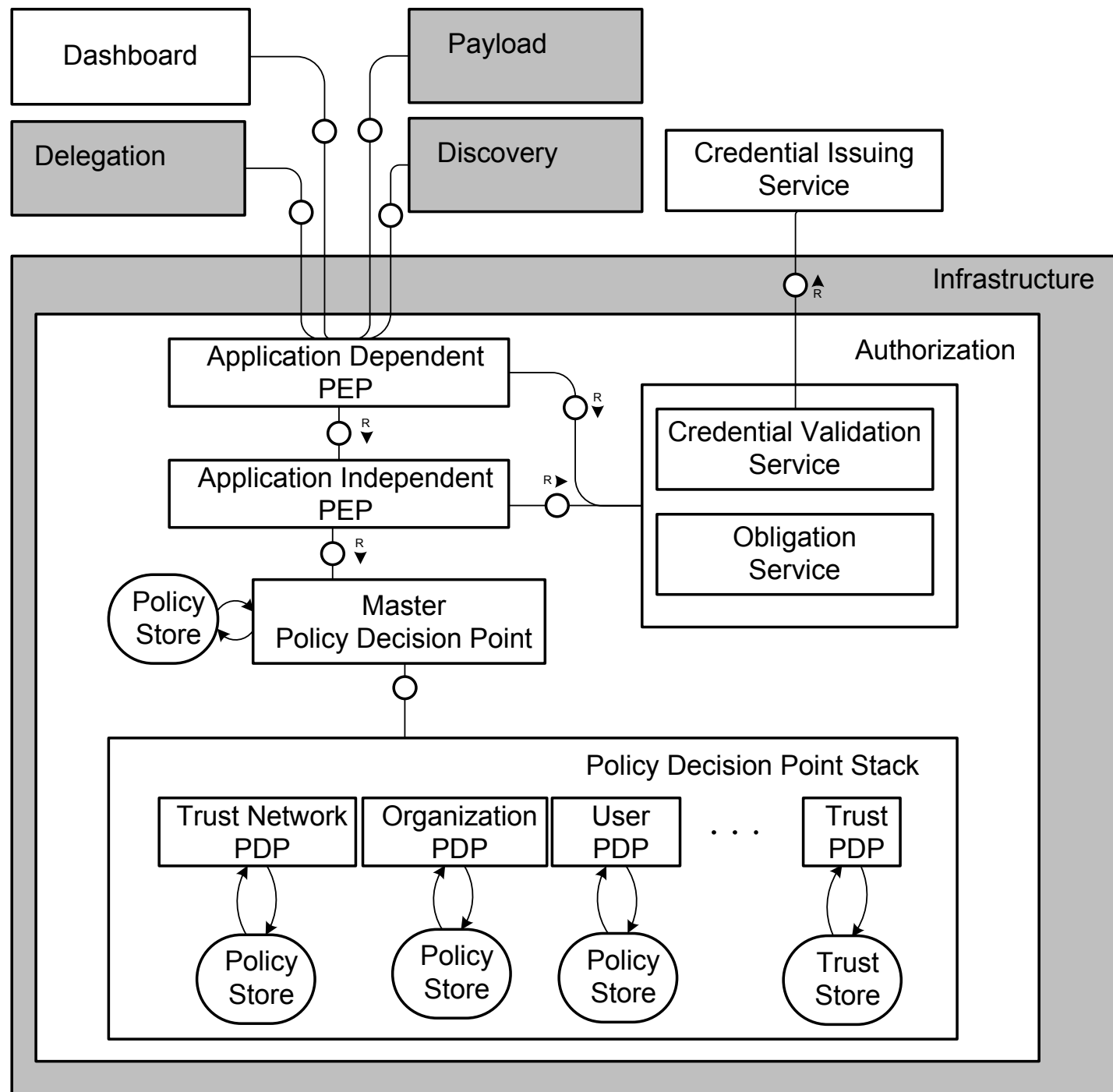












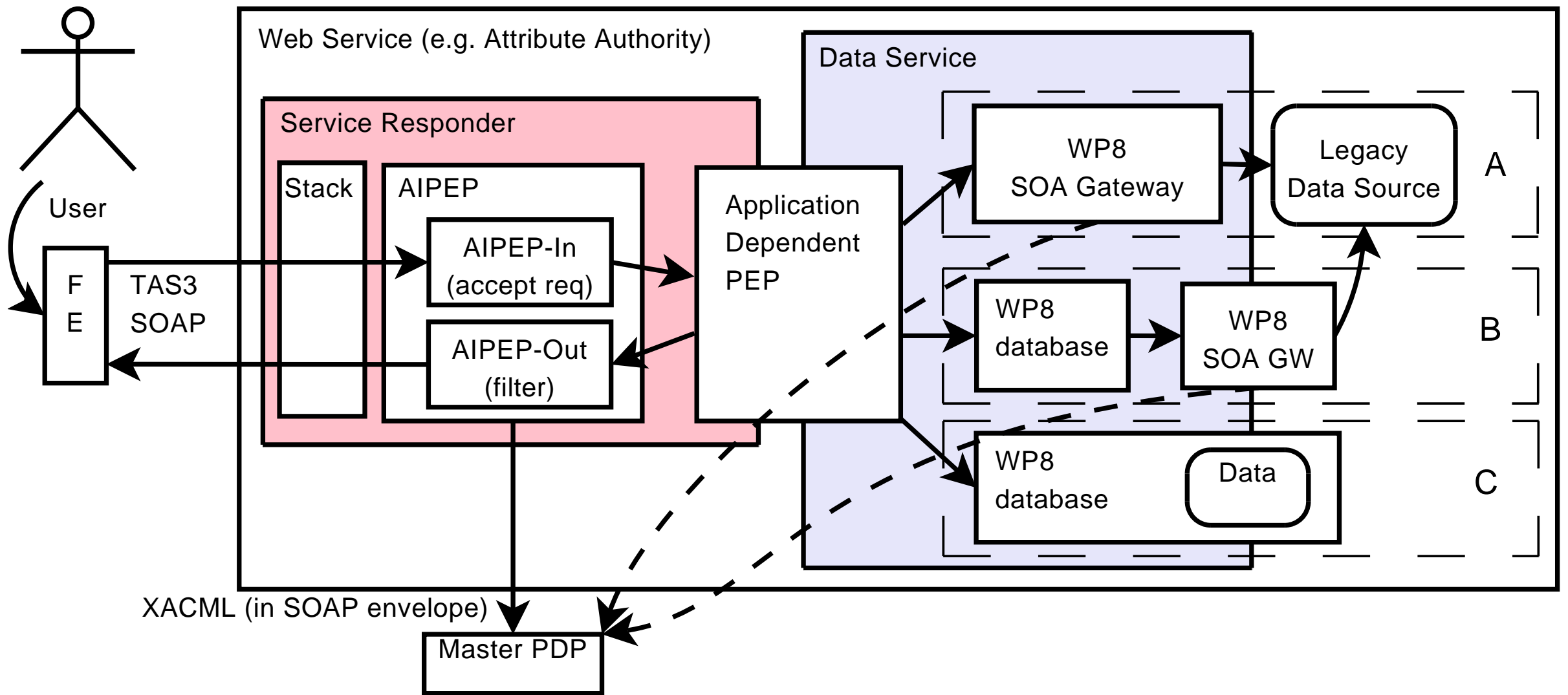


Figure 1: Application Integration using ADPEP and (A) Risarís SOA Gateway, (B) Fedora as frontend to Risarís, (C) Fedora database.

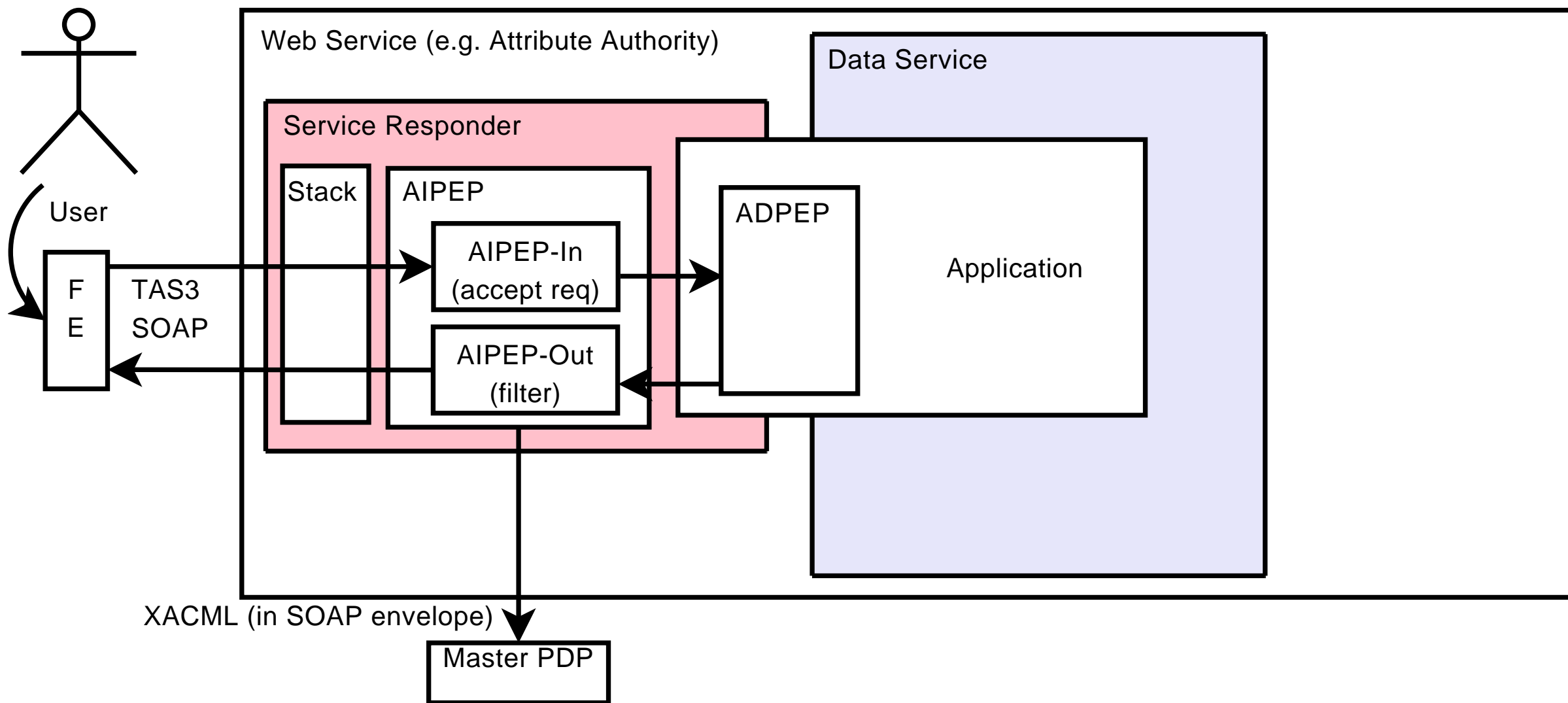


Figure 2: Application Integration: ADPEP implemented in application itself.

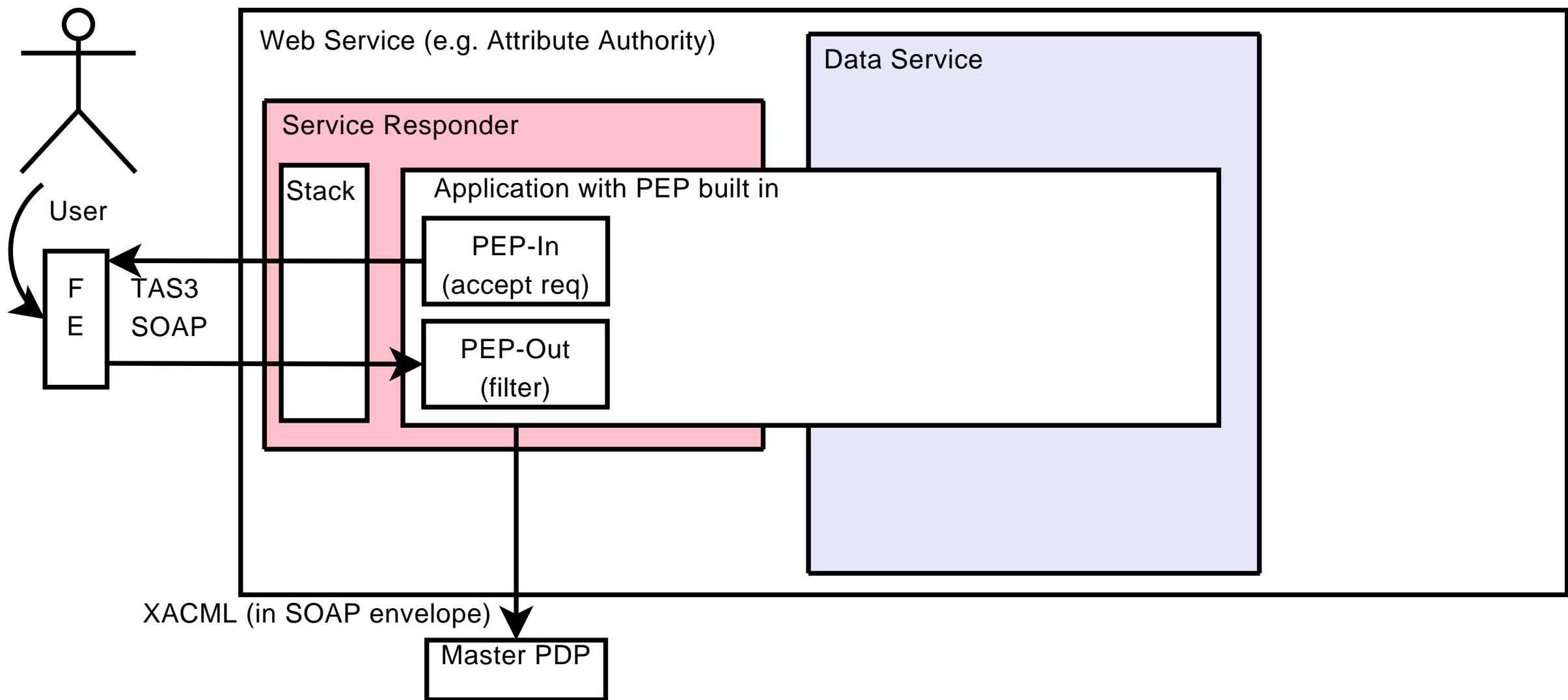
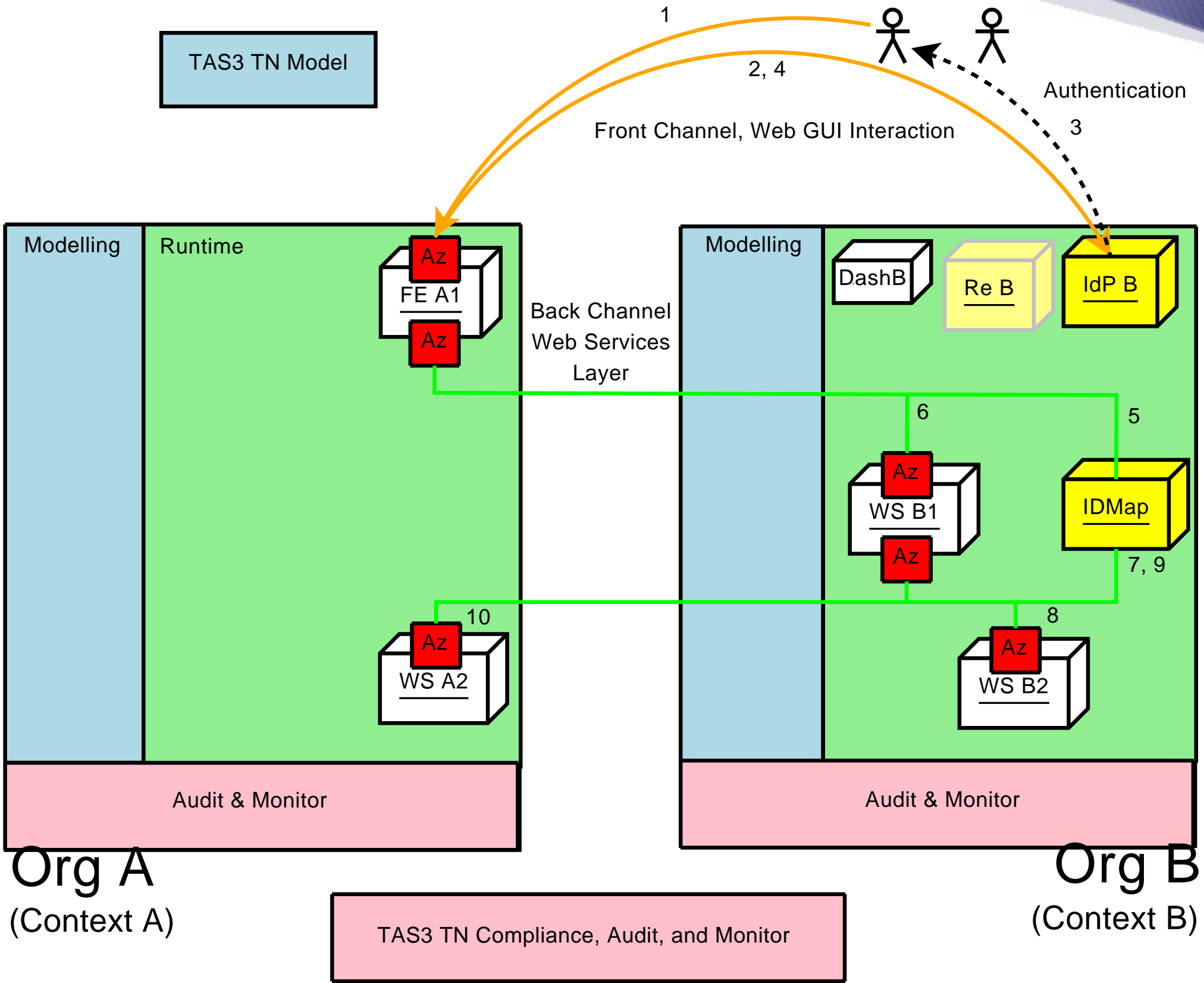
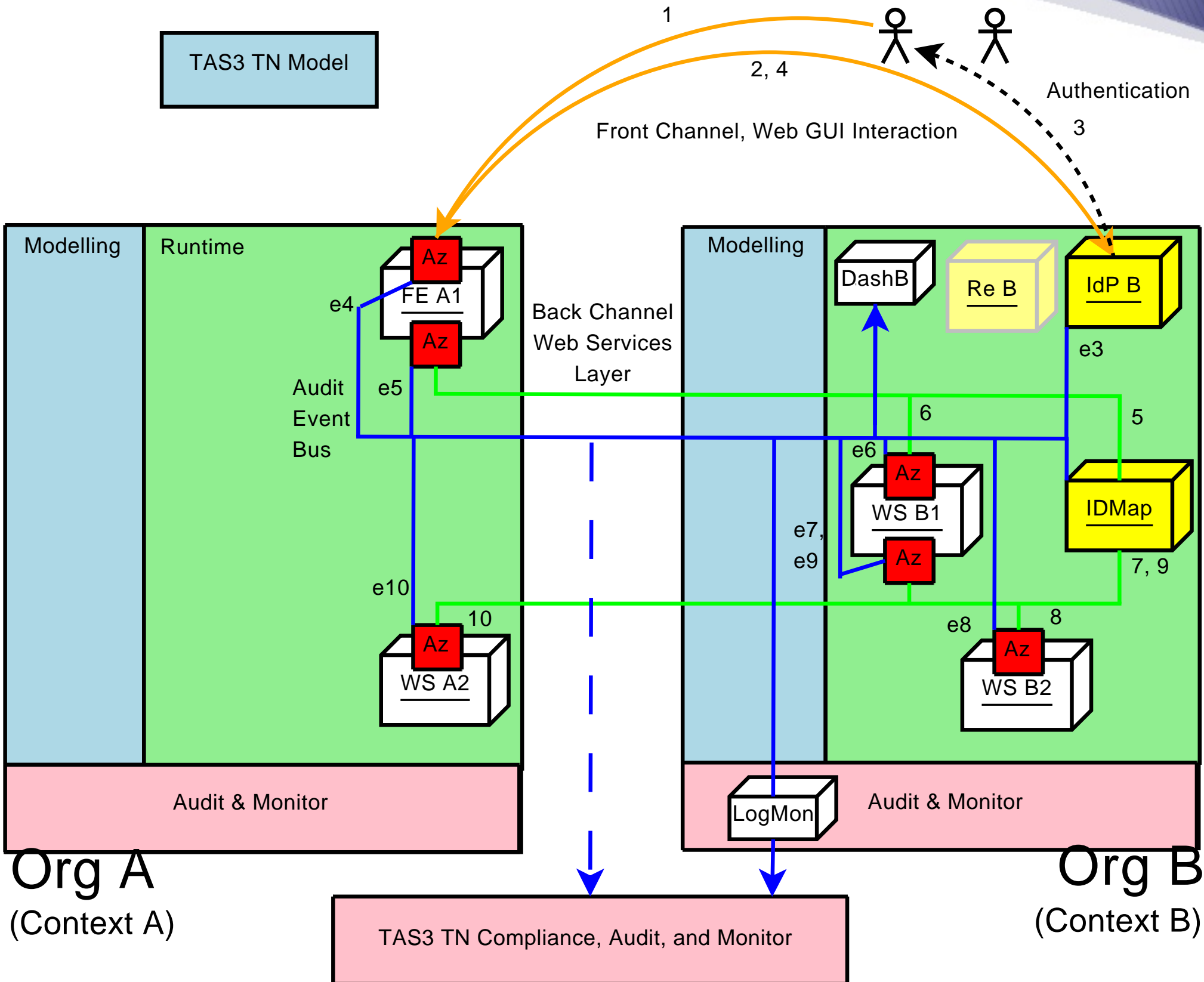


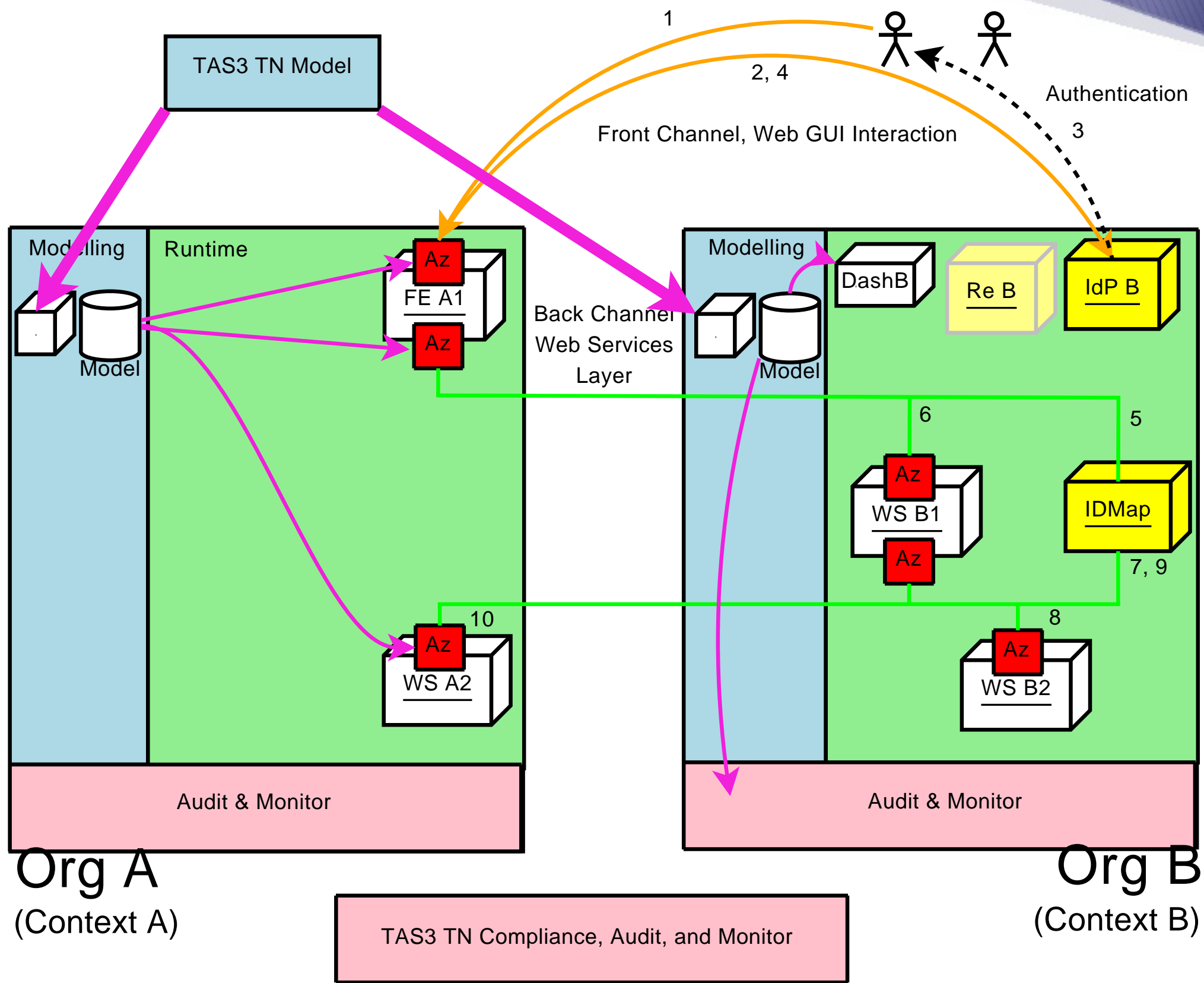
Figure 3: Application Integration: PEP implemented directly in application.

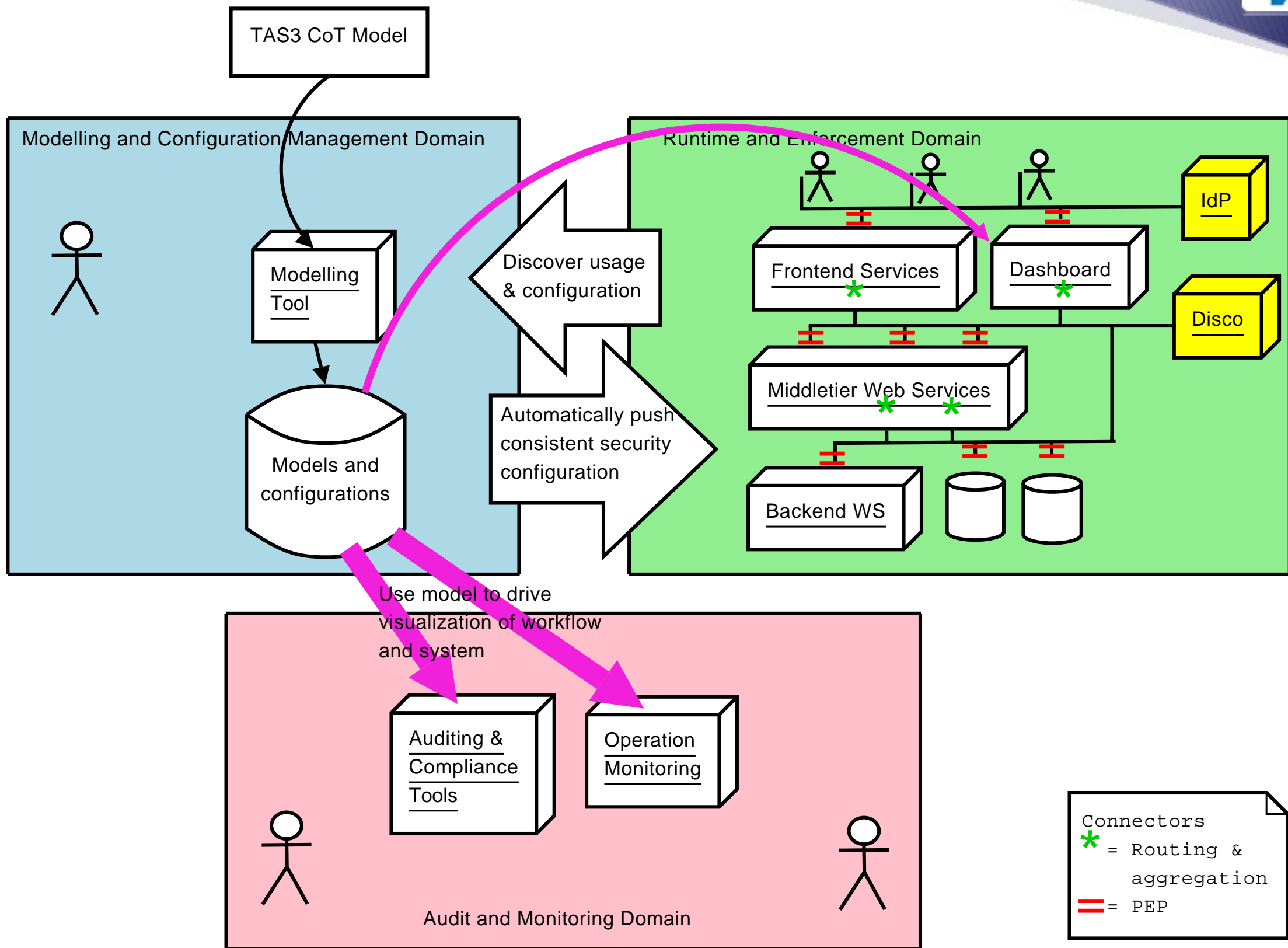
TAS3 TN Model

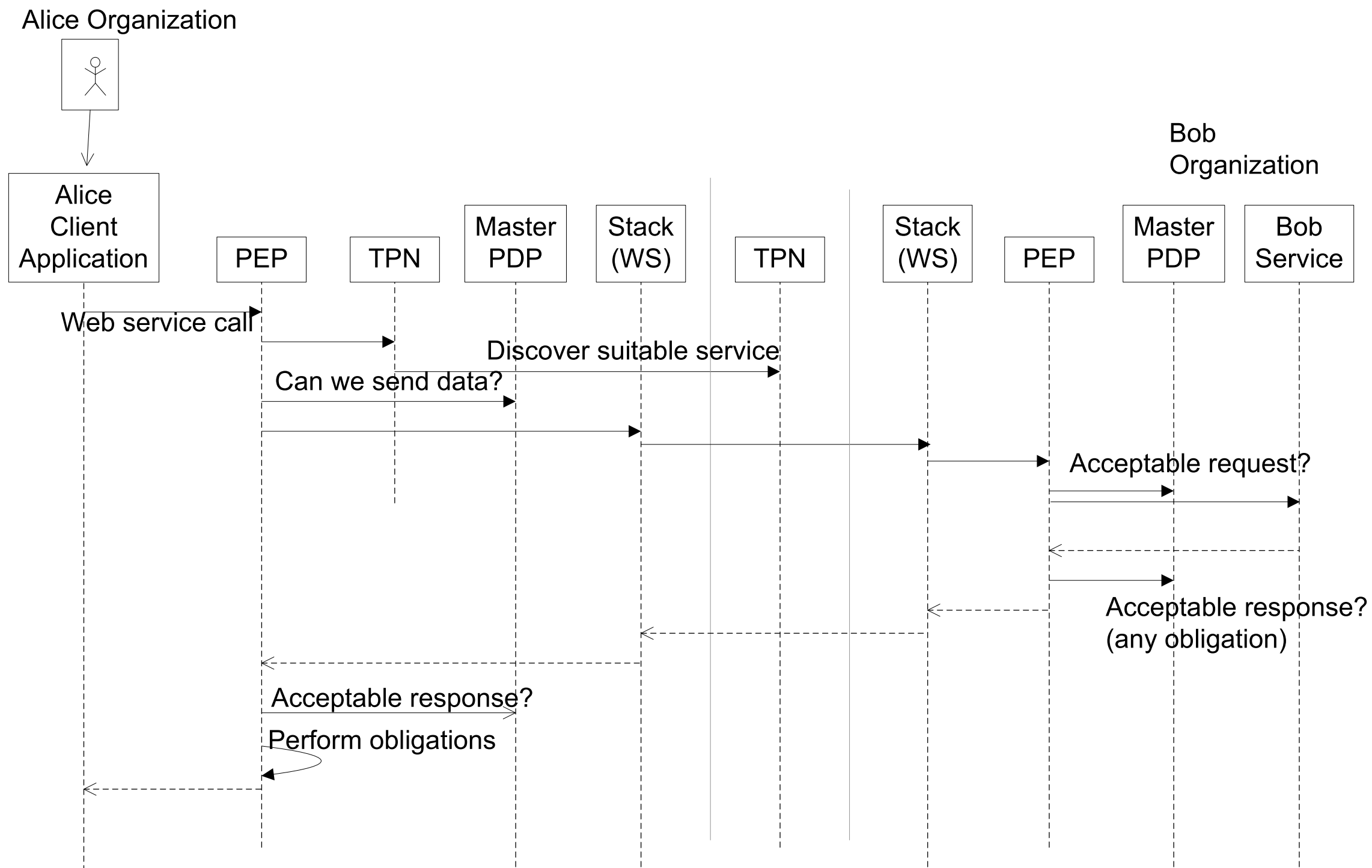


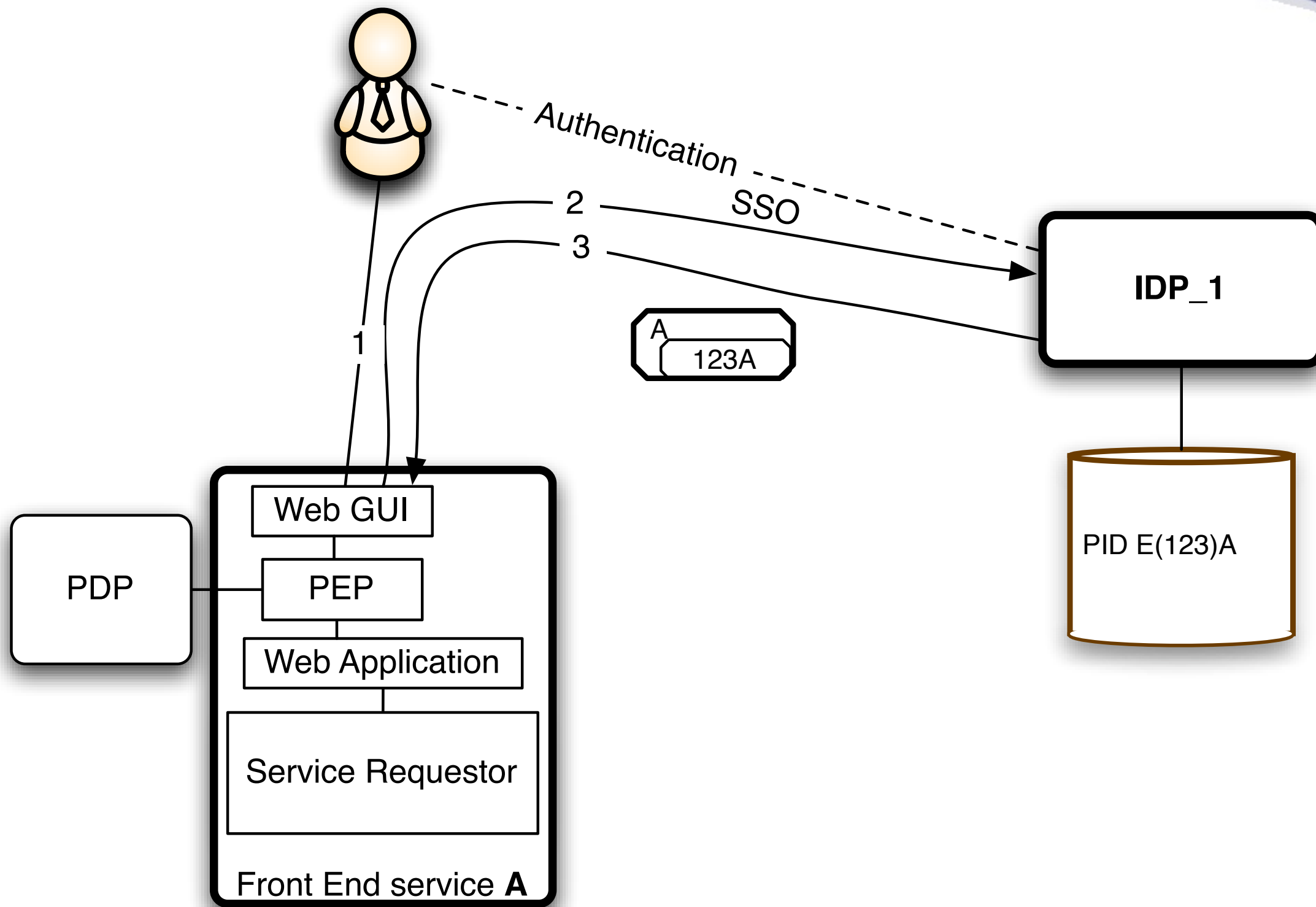
TAS3 TN Model

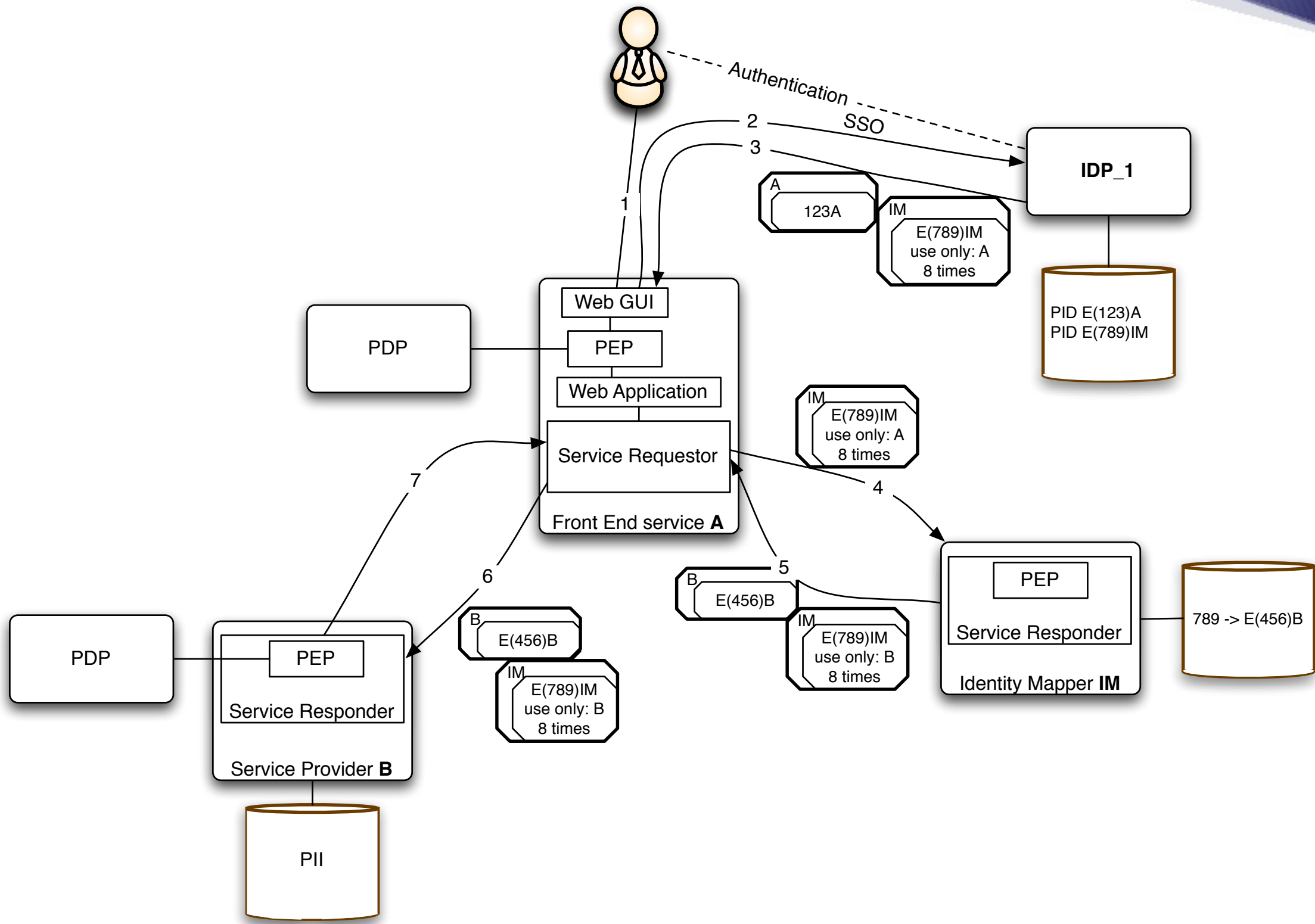


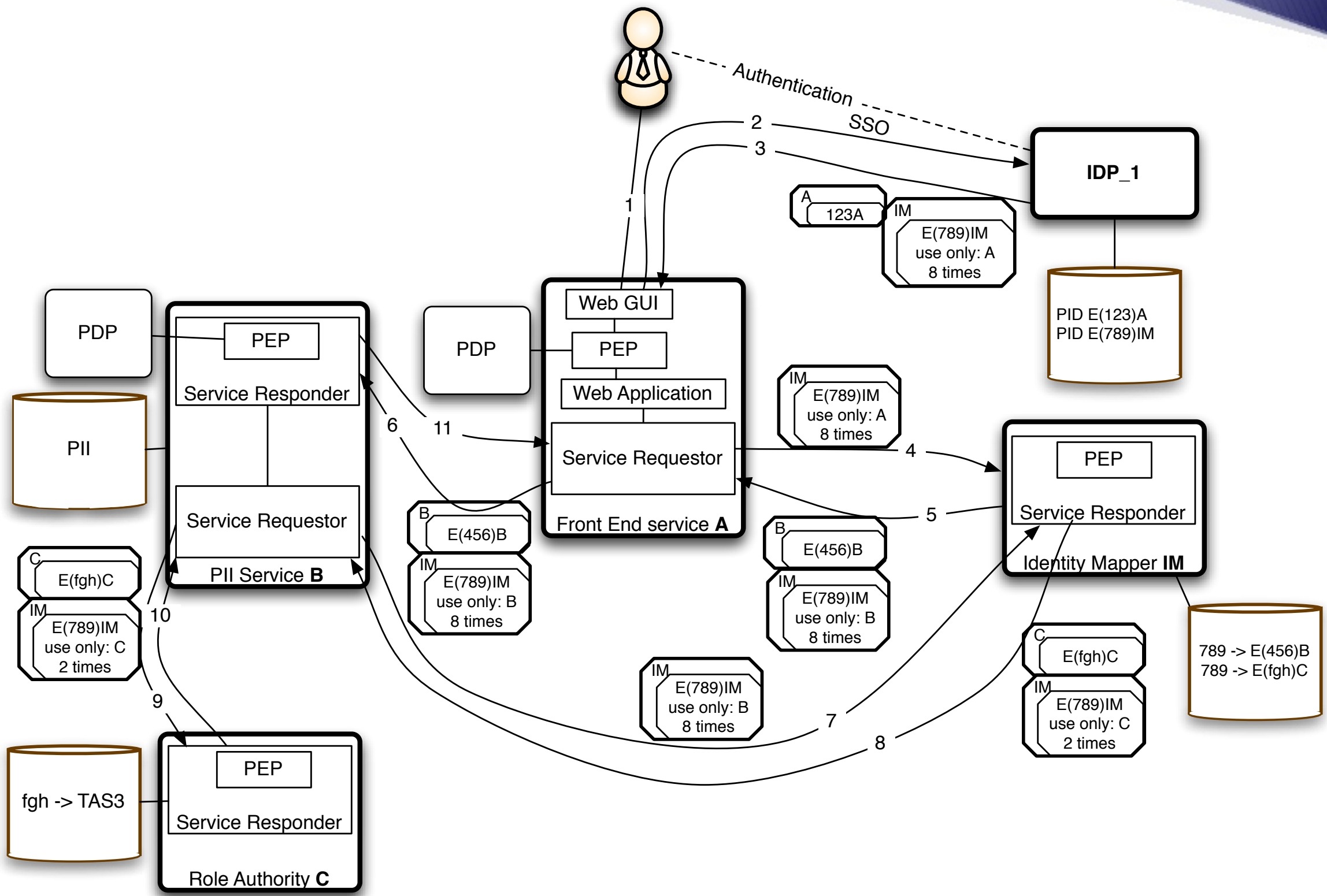


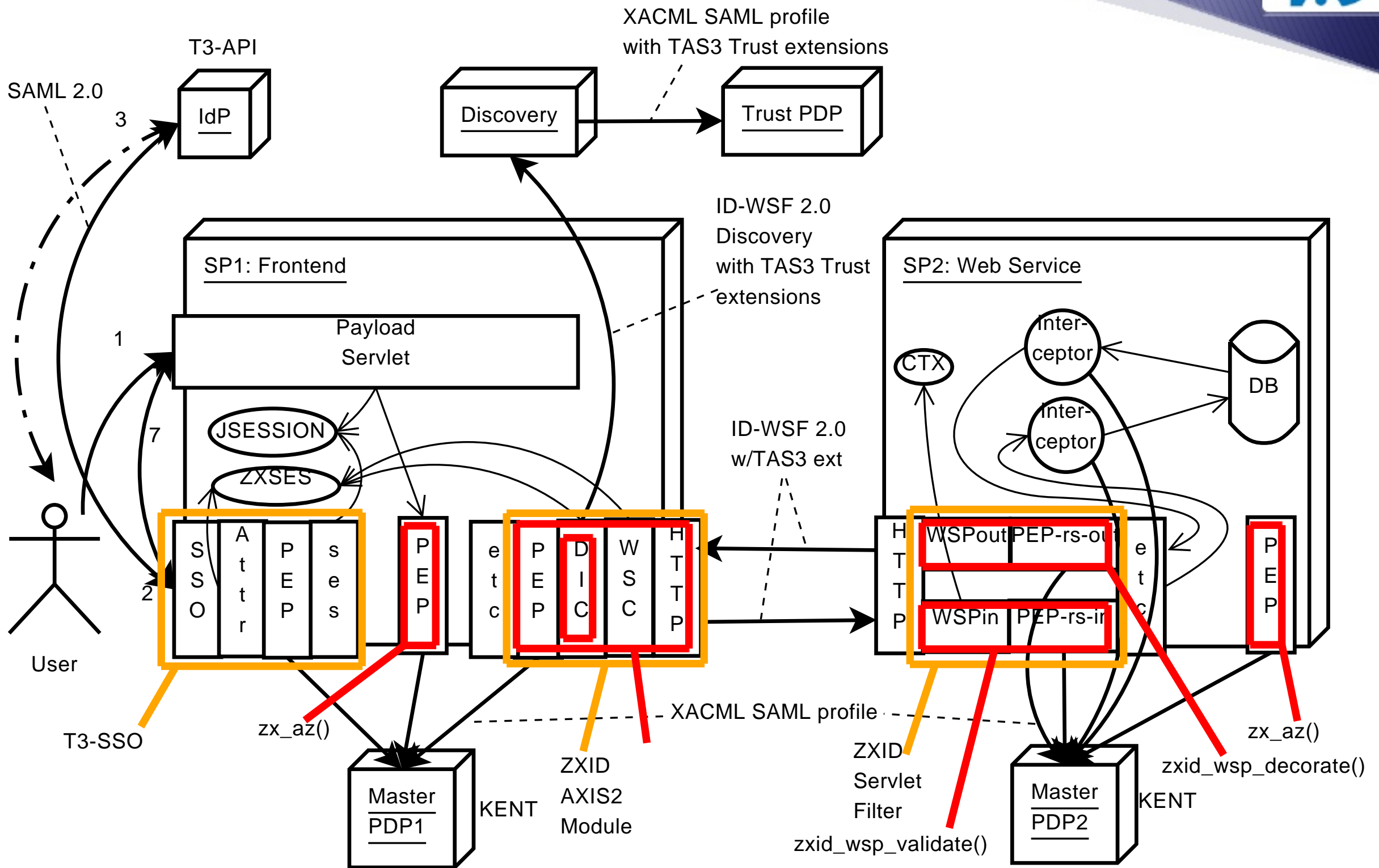












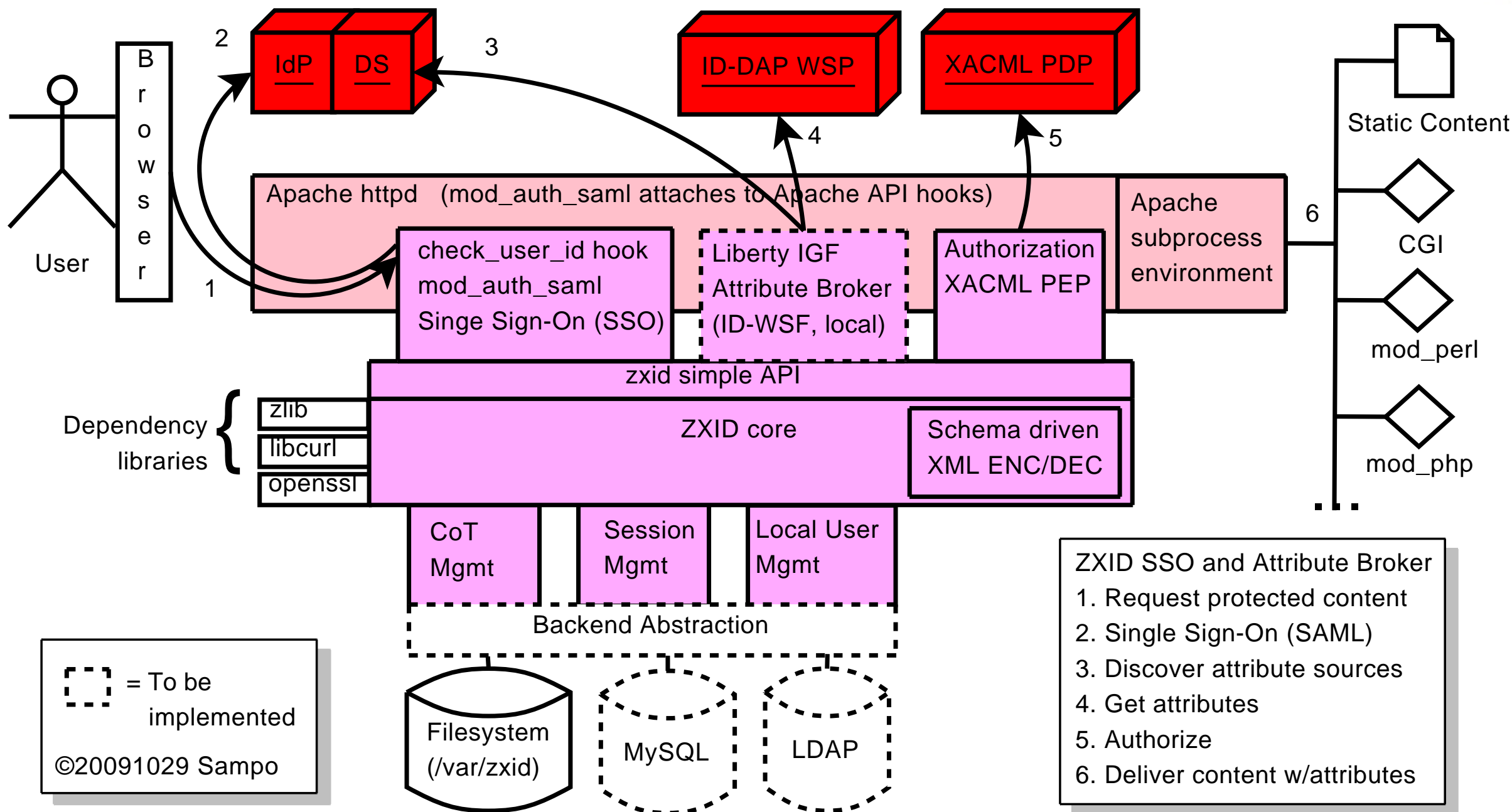


Figure 4: Software Architecture of `mod_auth_saml`.

TAS3 API

tas3_sso() SSO (with optional application independent authorization)

tas3_az() Application Dependent Authorization

tas3_call() Web Services Client: call a web service and validate response

tas3_wsp_validate() Validate that web service request can be processed

tas3_wsp_decorate() Create a web service response

TAS3 Using Java

```
01 import zxidjava.*;    // Pull in the zxidjni.az() API
02 import java.io.*;
03 import javax.servlet.*;
04 import javax.servlet.http.*;
05
06 public class zxidappdemo extends HttpServlet {
07     public void doGet(HttpServletRequest req, HttpSe
08         throws ServletException, IOException
09     {
10         String fullURL = req.getRequestURI();
11         if (req.getQueryString() != null)
12             fullURL += "?" + req.getQueryString();
13         System.err.print("Start ZXID App Demo GET("+ful
14         HttpSession ses = req.getSession(false); // Im
```

```
15     if (ses == null) {
16         res.sendRedirect("sso?o=E&fr=" + fullURL);
17         return;
18     }
19
20     res.setContentType("text/html");
21     res.getOutputStream().print("<title>ZXID Demo A");
22
23     // Render logout buttons (optional)
24
25     res.getOutputStream().print("[<a href=\"sso?gl=");
26
```

```
27 // The SSO servlet will have done one iteration
28 // serves to illustrate, how to explicitly call
29
30 if (zxidjni.az("PATH=/var/zxid/", "Action=Show"
31     res.getOutputStream().print("<p><b>Denied.<
32     //res.setStatus(302, "Denied");
33 } else {
34     res.getOutputStream().print("<p>Authorized.
35 }
36
37 // Render protected content page (your applicat
38
39 res.getOutputStream().print("<pre>HttpSession d
40 String[] val_names = ses.getValueNames();
41 for (int i = 0; i < val_names.length; ++i) {
```

```
42         res.getOutputStream().print(val_names[i] +
43     }
44
45     res.getOutputStream().print("</pre>");
46 }
47 }
```

TAS3 Performance

Public key equiv ops

ZXID Outline (1/2)

1 Create your own SP

1. Dummy using ZXID standalone code
2. Hookup to CoT, metadata
3. See it work
4. Integrate to your own code
5. See it work

2 Triggering Az from SSO

3 Using SSO attributes

4 Creating your own WSC

1. Demo of actual web service call, with traces
2. Integrating ZXID code to call existing service

ZXID Outline (2/2)

5 Providing your own WSP

1. Integrating ZXID code
2. Service Registration Step
3. Association Step
4. Making web service call: your WSC to your WSP

6 Interop

1. Discovering other people's WSPs
2. Your WSC calling other people's WSP
3. Your WSP being called by other people's WSC

7 mod_auth_saml tutorial

Thank You

Sampo Kellomäki (sampo@symlabs.com)

+351-918.731.007

Table 1: SAML and Liberty Open Source Implementations

| Product | License | Platform | SAML SP | SAML IdP | WSC | WSP | Disco | People | Interact | Account | Other |
|---------------|---------|-------------|---------|-----------|-----|-----|-------|--------|----------|---------|--------|
| ZXID.org | Apache2 | C + SWIG | Full | TBA | y | y | WSC | TBA | TBA | TBA | |
| mod_auth_saml | Apache2 | C + SWIG | Full | - | y | y | WSC | TBA | TBA | TBA | |
| Lasso | GPL2+ | C + SWIG | Cert | - | y? | ? | WSC | | | | |
| Authentic | GPL2+ | C + Python? | - | Certified | - | - | WSP | ? | ? | ? | |
| OpenSSO | Java? | pure PHP | Partial | - | - | - | - | - | - | - | |
| OpenSSO | Java? | Java | Cert | Cert | 1.1 | 1.1 | 1.1 | - | 1.1 | - | |
| OpenSAML | ? | Java? | Partial | Partial | | | | | | | |
| OpenLiberty | Apache2 | Java | - | - | y | - | WSC | TBA | - | - | AS-WSC |
| ConorCli | BSD? | C++ | - | - | y | - | WSC | WSC? | redir | ? | AS-WSC |
| ConorSvc | BSD? | Java | - | - | - | y | WSP | WSP? | redir | ? | AS-WSP |

TBA To be announced, on the road map, but not here yet

1.1 Only supports older ID-WSF 1.1 version of the services

C + SWIG C library with language bindings generated using SWIG

tool. SWIG supports among others C, C++, Perl, PHP, Python, Ruby, and Java language bindings. Generally open source products can be compiled for all popular operating systems such as Unix and Windows.

Additional info available on openliberty.org