DARIAH / DASISH AAI Workshop

Day 1: October 17, 2013

Developer Workshop on Shibboleth and SAML enabling Applications

Martin Haase, DAASI International

Martin.Haase@DAASI.de

DARIAH-EU

DAASI International

Agenda

- Welcome by the host (Laurence Horton, GESIS/DASISH)
- Overview and aim (Peter Gietz, DAASI/DARIAH)
- SAML and Shibboleth (Martin Haase, DAASI/DARIAH)
  - SAML profiles and flows
  - Shibboleth Identity Provider and Service Provider
- DARIAH Java SP (Tobias Gradl, UBamberg/DARIAH)
- Approaches for Shibbolizing applications (MH)
- CLARIN and Shibboleth: LAT (Willem Elbers, MPI.NL/CLARIN)
- PHP framework (MH)
- IdP discovery (Peter Schober, UVienna)
- DARIAH Authorization SP Blueprint (MH)
- Round table and discussions (Moderation: PG)

DARIAH-EU

DAASI International

Introduction to SAML

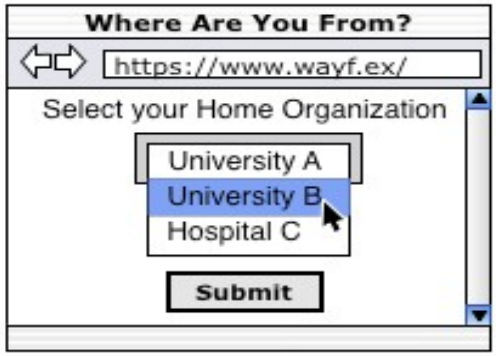
DARIAH-EU

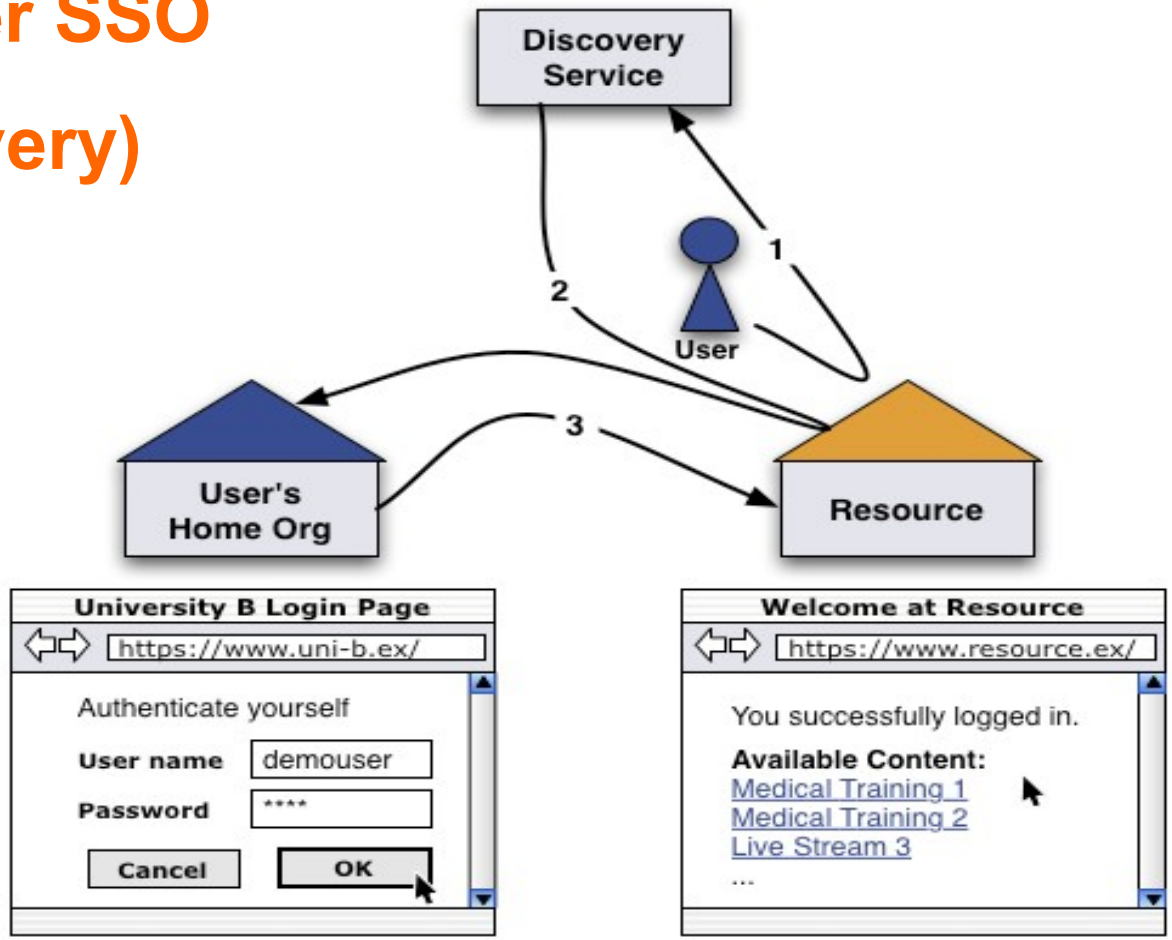
DAASI International

SAML Basics

- OASIS Standard, current version 2.0, March 2005
- Security Assertion Markup Language (SAML) specifies
  - Profiles (e.g. Web Browser SSO, Single Logout, Assertion Query, Attribute Usage)
  - Data exchange formats (esp. Assertions)
  - Protocols and Bindings
  - Metadata
- Components:
  - Identity Provider (IdP), lets users log in using the home organization's user directory
  - Service Provider (SP), protects Web resources and provides for information about the user sent by the IdP
  - IdP Discovery Service (DS, old term: Where-Are-You-From, WAYF)

SAML Profile: Web Browser SSO (with IdP Discovery)





DARIAH-EU

SAML Vocabulary

- Authentication, authN: proof a user's identity
- Authorization, authZ: grant the user access to a resource
- Single Sign-On, SSO: any mechanism enabling login without user interaction
- Authentication Request, issued by an SP for an IdP
- SAML Assertion, issued by an IdP for an SP
- EntityID: ID of an SP or IDP (an URL or URN)
- AA: Attribute Authority, Endpoint at IdP
- ACS: Assertion Consumer Service, Endpoint at the SP
- Session: Security context and cached data for a logged-in user (exists at both the IdP and the SP)
- SessionInitiator: Part of SP that generates SSO requests
- Artifact: A reference to an Assertion

DARIAH-EU

DAASI International

SAML Assertion

- „SAML Ticket“ issued to one single SP
- Usually signed and encrypted
- Can contain max. 3 statements:
  - 1 Authentication Statement: AuthN instant + method
  - 1 Attribute Statement: 1..n Attributes, each 1..n values
  - 1 Authorization Decision Statement
- Contains besides the three statements:
  - Issue Instant and Issuer EntityID
  - Signature by the IdP
  - Information about the authenticated Subject: Name Identifier (Format + Value), EntityID of IdP and SP
  - Conditions (optional), e.g. AudienceRestriction for the SP

SAML Profiles

- Web Browser SSO using various bindings (HTTP Redirect, HTTP POST, Artifact, SOAP)
- Enhanced Client or Proxy (ECP) for Browser-like Web Service Clients
- IdP Discovery
- Single Logout via Front- or Back-Channel
- Artifact Resolution
- Assertion Query / Request
- Various other:
  - Name Identitfier Management
  - Name Identifier Mapping
  - Attribute Naming Formats

SAML further Core Concepts

- Protocols
  - Abstract form: Request / Response
  - Usually one for each of the mentioned profiles
- XML Signature
- XML Encryption
- Bindings: method how a message is transported
  - HTTP Redirect
  - HTTP POST (and HTTP-POST-SimpleSign)
  - SOAP and reverse SOAP (PAOS)
  - „HTTP Artifact“ (plus SOAP)

Base for SP ↔ IdP interaction

- Metadata
  - SPs find information about IdPs, e.g. various endpoint locations for each binding, e.g. the SSO Service
  - IdPs find informationen about SPs, e.g. the ACS
  - embedded public X.509 keys for signature and encryption
  - various extensions (logo urls, contact info, Attr needs)
  - Metadata are public and usually signed
- Common Attribute encoding format, e.g.
  - urn:oid:2.5.4.42 for givenName
  - urn:oid:1.3.6.1.4.1.10126.1.35.3.15 for „TGacceptedTermsOfUse“ (used in TextGrid)
- Synchronized clocks, HTTPS, etc...

DARIAH-EU

DAASI International

Federation

- Federation concept: a trust infrastructure of IdPs and SPs („circle of trust“)
- Minimal tasks:
  - Metadata maintenance (registration, access, signature)
  - Discovery Service operation
  - Legal issues...
- IdPs and SPs can be members of multiple federations (Federation ↔ Metadata)
- An IdP and an SP can trust each other also without a federation: just exchange metadata bilaterally
- Some SP / IdP implementations even cannot handle SAML metadata: just exchange some information that is contained in metadata

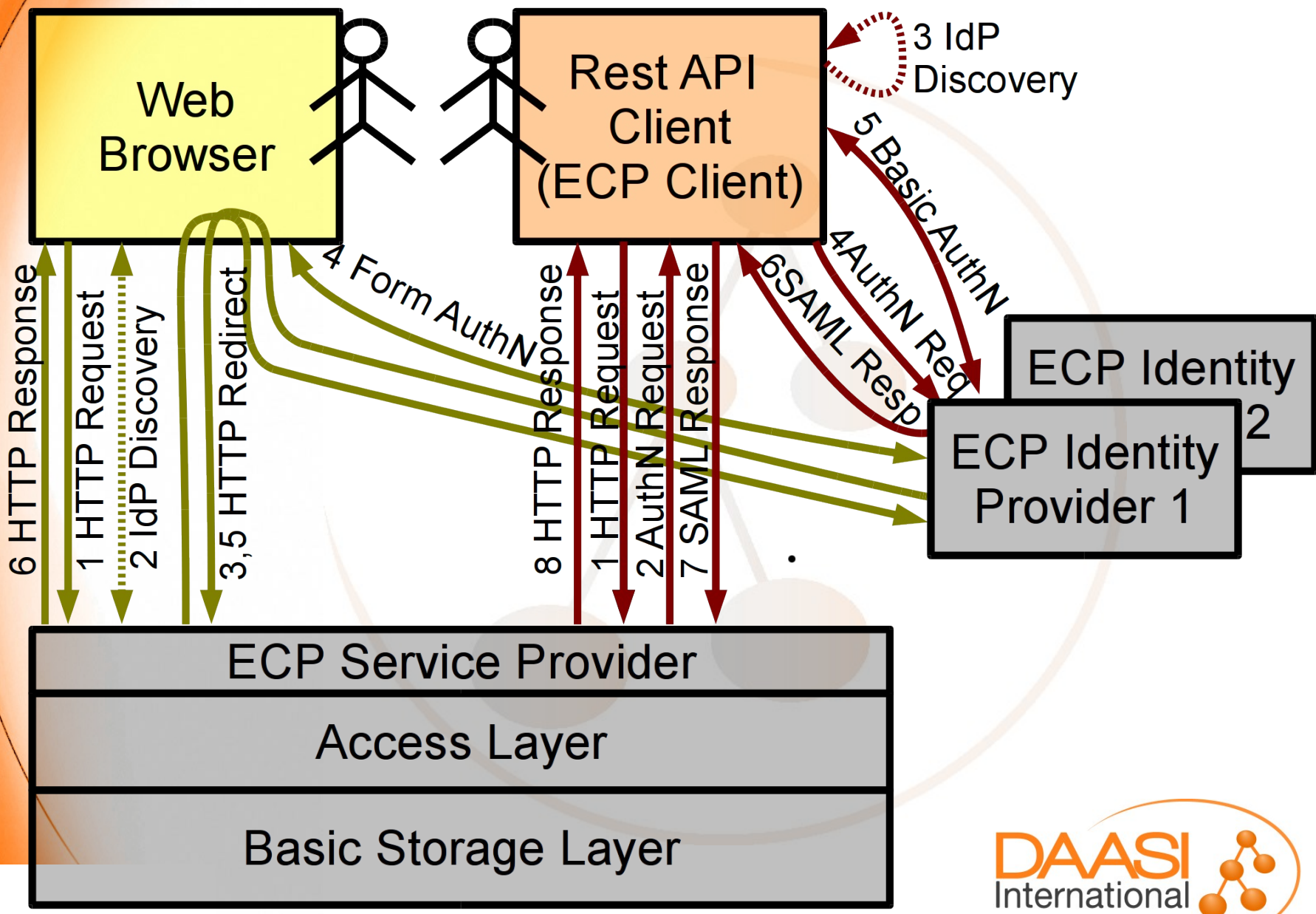
DARIAH-EU

DAASI International

WebSSO vs. ECP

- Different SAML SSO Profiles:
  - WebSSO for Web Browser and usual Web Applications
  - ECP (Enhanced Client or Proxy) for other clients not capable of displaying a login page → can be used for RESTlike Web Services
- ECP needs Clients with special, Browser-like features
  - Issue HTTP GET, POST requests and follow HTTP Redirects
  - Process HTTP Headers
  - Handle Cookies
  - Process XML (SOAP + SAML messages themselves)
  - Handle HTTPS
- Supported by Shibboleth IdP (only Basic Auth) and SP

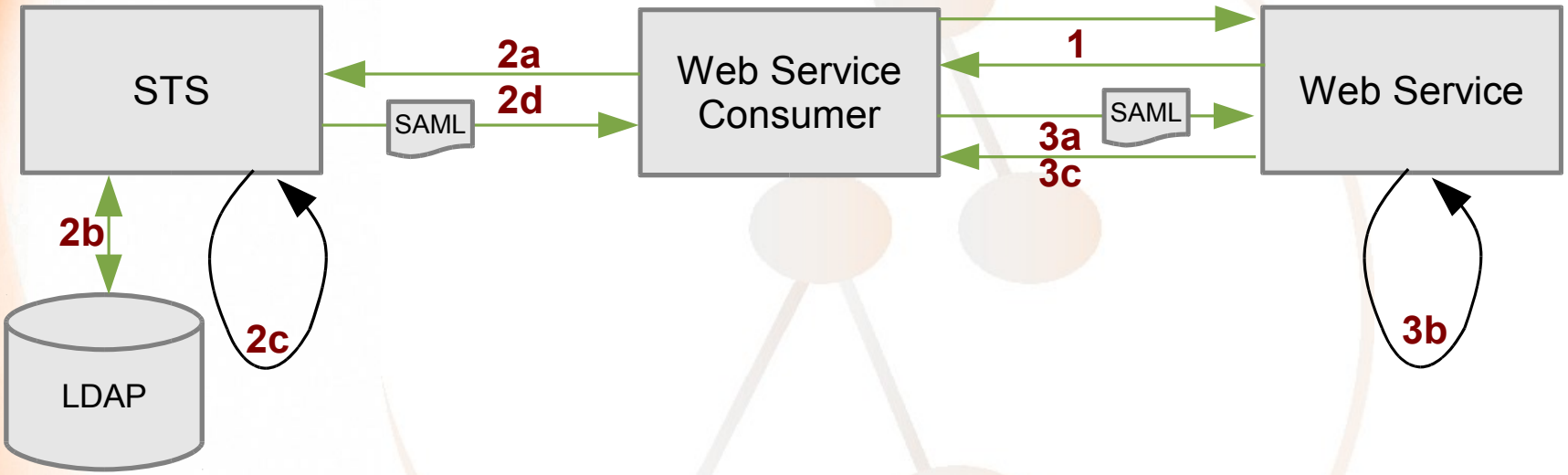
WebSSO vs. ECP



WS-Security and SAML

- Using SAML Assertions as WSS-Tokens for SOAP Web Services
- The Assertion is embedded in the SOAP Header just like another Authentication Token
- WSS SAML Token Profile defines this model
- IdP = Security Token Service (STS)
- SP issues a RequestSecurityToken (RST) message via the Client for the STS (RST is defined in WS-Trust)
- STS reply: RequestSecurityTokenResponse (RSTR)
- NOT supported by Shibboleth

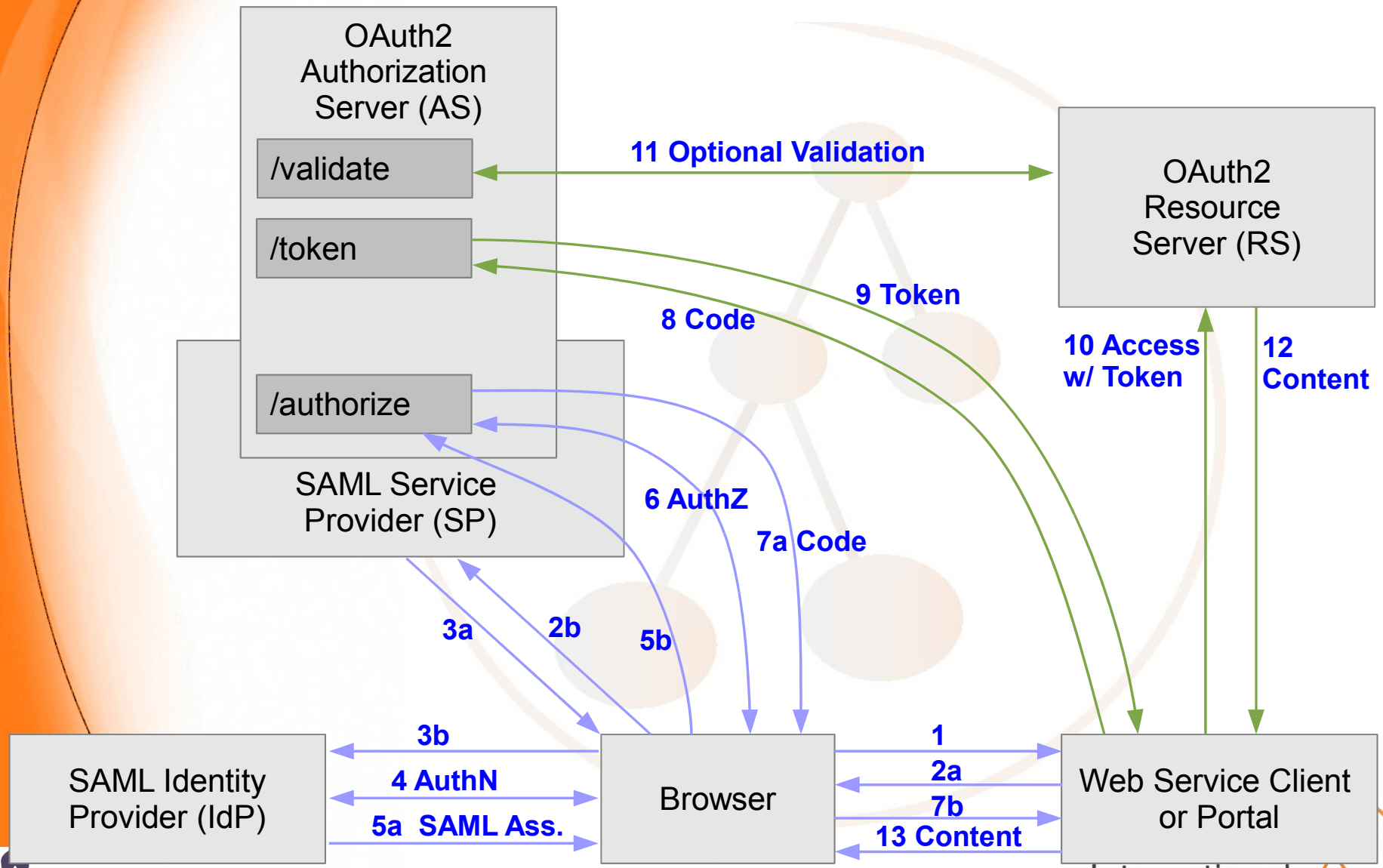
WS-Security and SAML



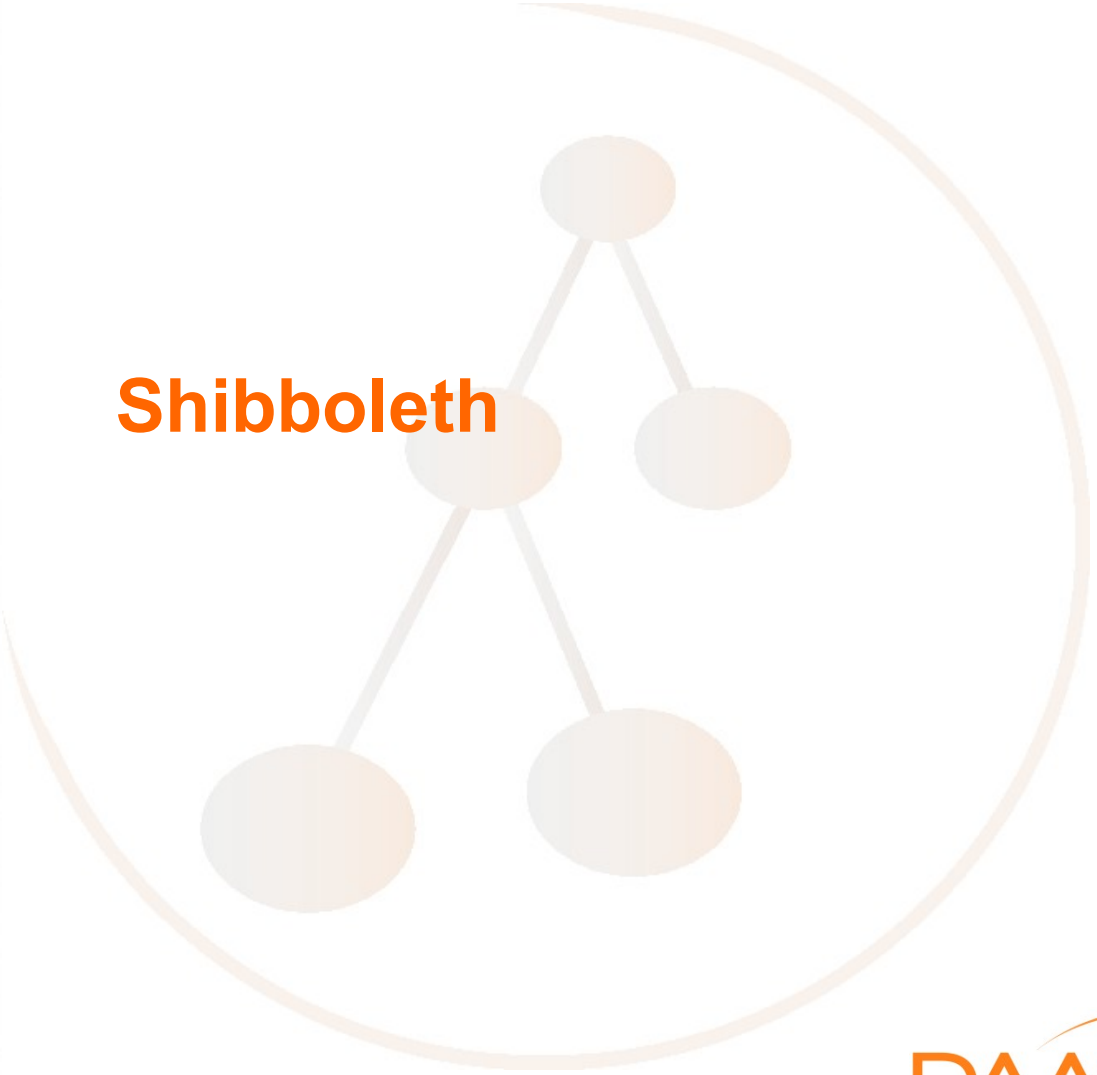
Web Services Delegation

- Delegation
  - 1-Tier: IdP issues „token“ for User U to Service A and A requests resources from Service B on behalf of U
  - N-Tier: Service A → Service B → Service C→ ...
- Delegation with SAML/Shibboleth alone is complex (https://spaces.internet2.edu/display/ShibuPortal/Home)
- OAuth2.0 allows and is designed for 1-Tier delegation
- Combination of SAML infrastructure and OAuth2.0 is possible
- Most common use case: some SAML-protected Web portal application wants to access some RESTlike service in the user's name

Web Services Delegation



Shibboleth



Origin

- Open-Source project, originally developed by Internet2
- Now managed by the Shibboleth Consortium
- Implementation of
  - SAML Identity Provider
  - SAML Service Provider
  - SAML Discovery Service (Centralized and Embedded)

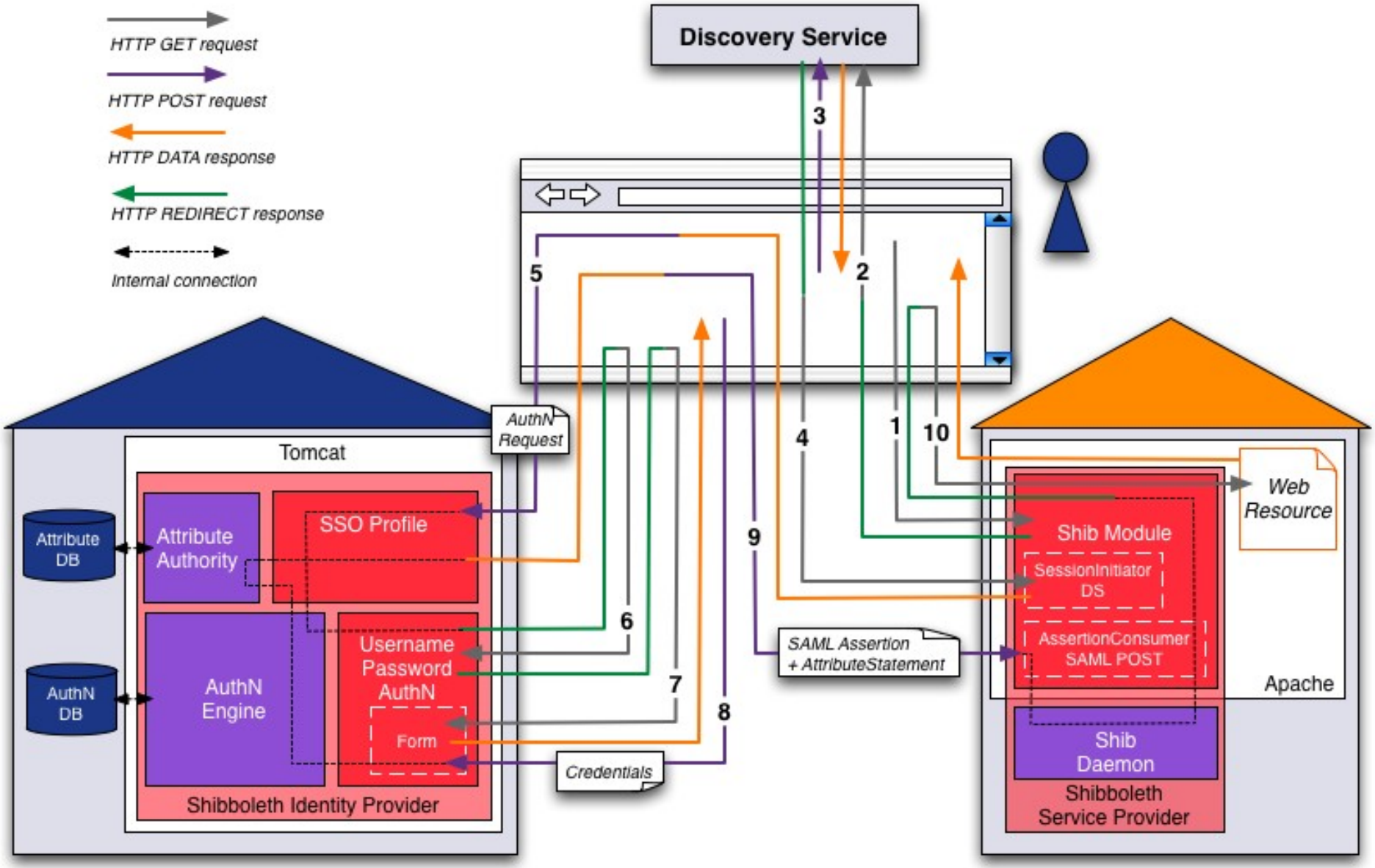
Shibboleth®

- Origin of the word is Hebrew, see the Bible, Judges 12,6
- Current version 2 implements SAML2 and is compatible with SAML1

DARIAH-EU

DAASI International

Shibboleth WebSSO



Identity Provider

DARIAH-EU

DAASI International

IdP Basics

- IdP provides users with the possibility to use Web applications that are protected by trusted SPs
- Unlimited number of “connected” SPs
- An Organization's IdP can be used for
  - Organization internal Single Sign-On
  - SSO within a federation
  - SSO in multiple federations
- Authentification of a user is never done at the SP, but always locally in the home organization that the IdP represents → the password is only there!

DARIAH-EU

DAASI International

IdP Basics (2)

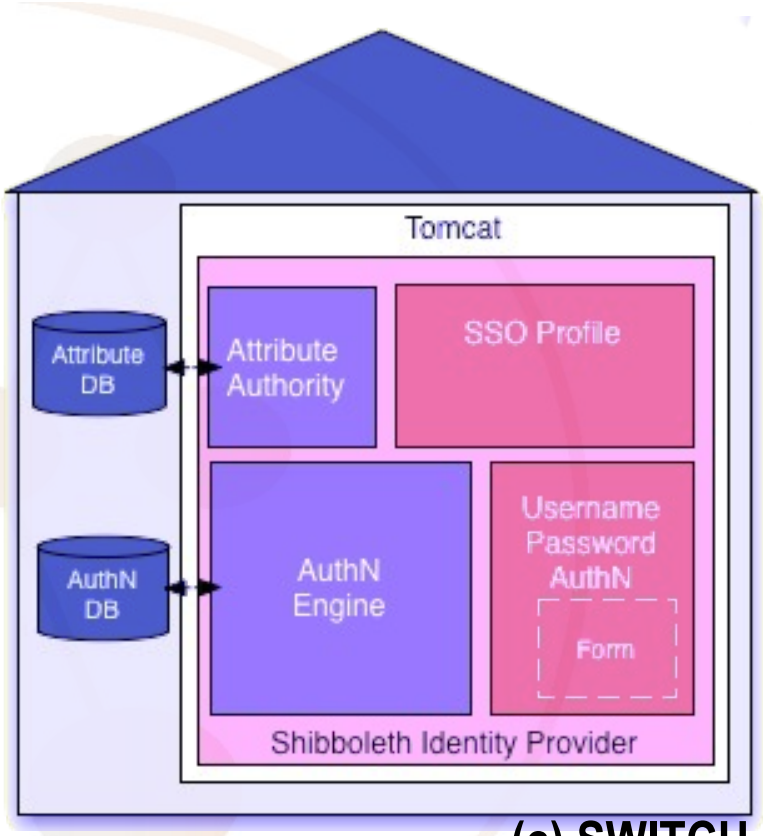
- After a successful authentication, the IdP sends a SAML Token / Ticket (*Assertion*) to the SP, authenticating the user
- A reference to the IdP session is saved in the user's Web browser
- Thanks to the IdP session, any SP can be issued a new Assertion (within session lifetime)
- Users will not realize following log-ins to further SPs (Single Sign-On) as they are automatically redirected to the IdP (HTTP Redirect, POST) that keeps the session

DARIAH-EU

DAASI International

IdP Architecture

- IdP is a Java Web application
- Installation on Windows, Linux, Unix, ...
- Runs in Servlet container (Tomcat, Jetty)
- Possibly Apache Web server as proxy
- Authentication and attribute resolution are separated
- Extensible (Spring-based configuration)



(c) SWITCH

Connection to the IdM

- Authentification: IdP can use
  - LDAP, AD, relational Database via JAAS (username/password form)
  - Kerberos domain
  - External SSO (CAS, Pubcookie etc)
  - Remote-User based via the container (Apache, Tomcat)
- Attribute sources: various DataConnectors
  - LDAP, AD, relational Database
  - static Attributes
  - Stored Ids (for pseudonyms) from relational DB

DARIAH-EU

DAASI International

Example: connect LDAP via JAAS and Username/Password form

- JAAS configuration

```
edu.vt.middleware.ldap.jaas.LdapLoginModule required
   ldapUrl="ldap://localhost:389"
   baseDn="dc=dariah,dc=eu"
   tls="true"
   subtreeSearch="true"
   serviceUser="cn=manager,dc=dariah,dc=eu"
   serviceCredential="secret"
   userFilter="uid={0}";
```

- Many more options here, e.g.
  - LDAP Failover,
  - LDAPS
  - Authorization filters
- Other JAAS modules analogous

Interplay with SPs

- Attribute resolver reads attributes from directories, databases, etc. and defines their format on the wire
  - Attributes should be based on standards: e.g. eduPerson eduPersonAffiliation, eduPersonTargetedId, eduPersonEntitlement
  - A particular IdP and SP can arrange bilaterally for custom attributes, additionally
- Attribute filter defines Attribute Release Policies
  - Shibboleth provides for fine-granular attribute filtering rules (data protection)
  - Complex filters possible

Example: Attribute Resolver with an LDAP DataConnector

```
<resolver:DataConnector id="myLDAP"
    xsi:type="dc:LDAPDirectory"
    ldapURL="ldap://localhost:389" useStartTLS="true"
    baseDN="dc=dariah,dc=eu"
    principal="cn=manager,dc=dariah,dc=eu"
    principalCredential="secret">
    <dc:FilterTemplate>
        <![CDATA[
            (uid=$requestContext.principalName)
        ]]>
    </dc:FilterTemplate>
    <dc:ReturnAttributes>
         entryDN mail uid cn sn givenname
    </dc:ReturnAttributes>
</resolver:DataConnector>
```

Example: Attribute Definition

- also in attribute-resolver.xml

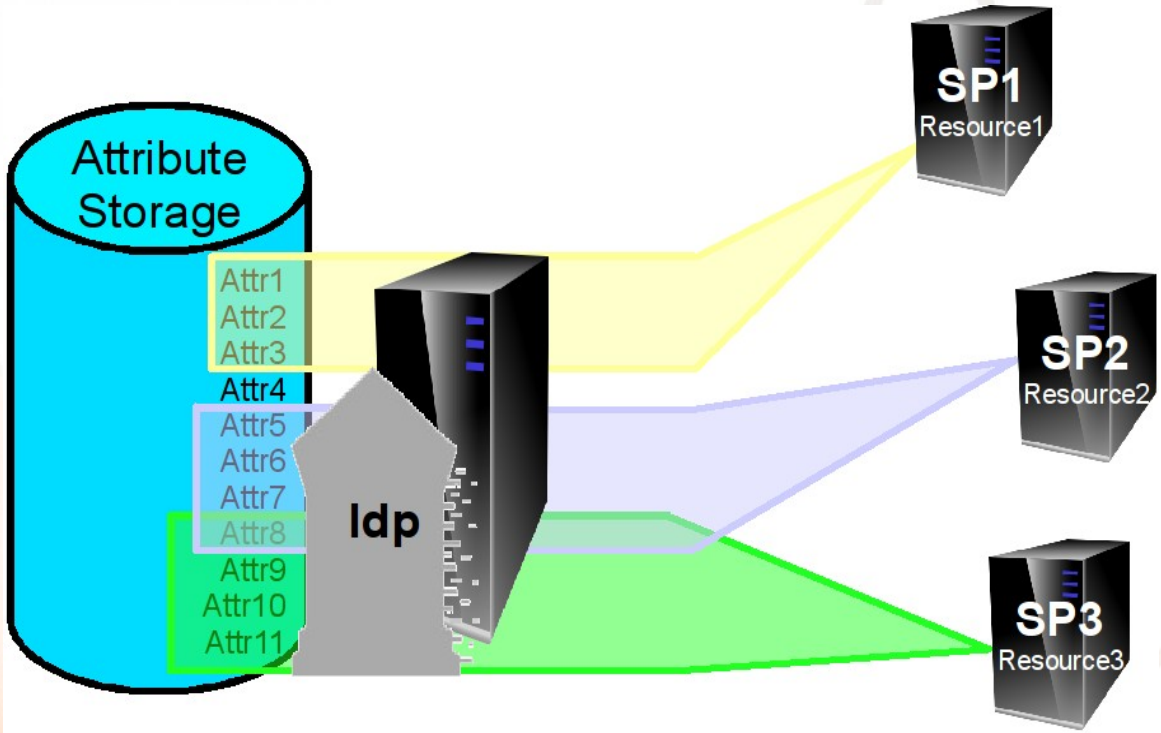
```
<resolver:AttributeDefinition xsi:type="ad:Simple"
          id="email"
          sourceAttributeID="mail">
     <resolver:Dependency ref="myLDAP" />
     <resolver:AttributeEncoder xsi:type="enc:SAML1String"
          name="urn:mace:dir:attribute-def:mail" />
     <resolver:AttributeEncoder xsi:type="enc:SAML2String"
          name="urn:oid:0.9.2342.19200300.100.1.3"
          friendlyName="mail" />
</resolver:AttributeDefinition>
```

Example: Custom Attribute Definition

```
<resolver:AttributeDefinition xsi:type="ad:Mapped"
          id="fooBarRole"
          sourceAttributeID="entryDN">
    <resolver:Dependency ref="myLDAP" />
    <!-- no SAML1 Attribute Encoder necessary, this is a
          SAML2-only deployment -->
    <resolver:AttributeEncoder xsi:type="enc:SAML2String"
              name="urn:oid:1.2.3.4.5.6.7.8.9.1"
              friendlyName="fooBarRole" />
    <ad:ValueMap>
      <ad:ReturnValue>$1</ad:ReturnValue>
      <ad:SourceValue>[^,]+,ou=([^,]+),.*</ad:SourceValue>
      <!-- SourceValue contains user type in DN -->
    </ad:ValueMap>
</resolver:AttributeDefinition>
```

Attribut-Filter per SP

- Specify which attributes are sent to which SP
- Data sparseness
- Configure access rules centrally
- Allows for uniform SP configurations



Example: Attribute Filter

```
<afp:AttributeFilterPolicy>
    <afp:PolicyRequirementRule
          xsi:type="basic:AttributeRequesterString"
          value="https://sp1.example.eu/shibboleth" />

  <afp:AttributeRule attributeID="givenName">
        <afp:PermitValueRule xsi:type="basic:ANY" />
  </afp:AttributeRule>

  <AttributeRule attributeID="fooBarRole">
     <PermitValueRule xsi:type="basic:OR">
       <basic:Rule
          xsi:type="basic:AttributeValueString"
          value="Researchers"/>
       <basic:Rule
          xsi:type="basic:AttributeValueString"
          value="Students"/>
     </PermitValueRule>
  </AttributeRule>
</afp:AttributeFilterPolicy>
```

IdP Deployment Options

- Standalone Servlet Container or proxied by Apache Web Server?
- Jetty? Which version?
- Tomcat? Which version?
- Which Java Version?
- Clustering?
  - Do not cluster: active and standby instance (needs Load Balancer)
  - Parallel IdPs (stateless Clustering, with a LB and long cookie-based stickyness for SSO), no Artifact and SLO
  - Clustering with Terracotta (using DNS Round Robin or a LB)
  - Clustering with memcached (with LB, short stickyness)

DARIAH-EU

DAASI International

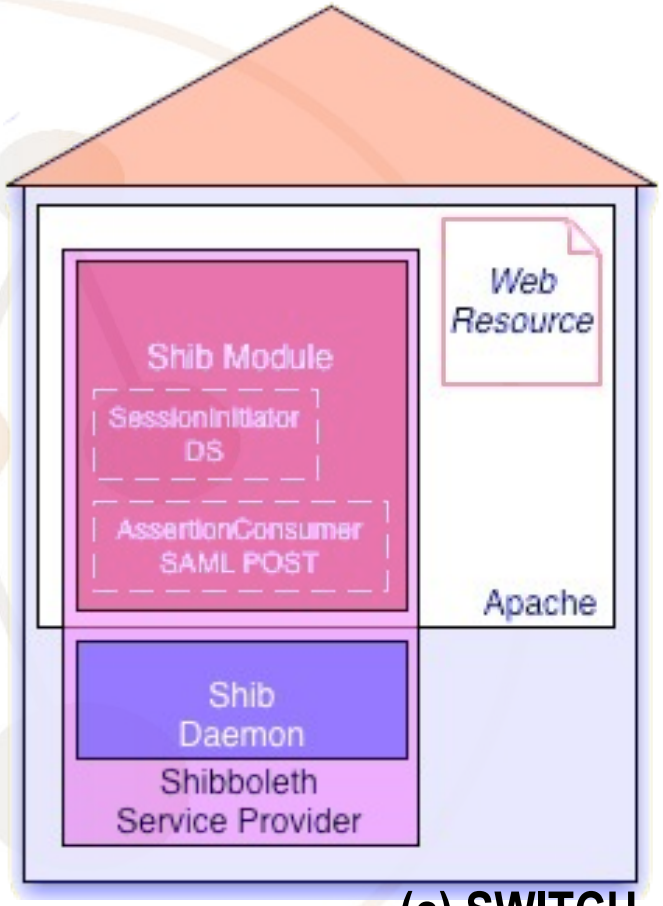
Service Provider

DARIAH-EU

DAASI International

Service Provider

- Supports Apache, FastCGI, Sun/iPlanet and IIS
- Application is protected by a Web server module
- Status (sessions, etc) is being kept by the Shibboleth Daemon shibd
- Binaries for Windows, RPM-based Linuxes, OS X, Sources for other UNIXes



(c) SWITCH

SP Configuration files

- Apache /etc/httpd/conf/httpd.conf, or in extra files (example for CentOS)
  - conf.d/<servername>.conf
  - conf.d/shib.conf, refernced therein mod_shib_22.so
- Shibboleth itself under /etc/shibboleth/
  - shibboleth2.xml
  - attribute-policy.xml
  - attribute-map.xml
  - xxx-metadata.xml
  - certificates
- Start skript /etc/init.d/shibd

Configure vhost

```
...
Listen 443
<VirtualHost _default_:443>

    ServerName          sp123.example.edu
    UseCanonicalName    On

    SSLEngine on
    SSLCipherSuite ALL
    SSLOptions +StdEnvVars +ExportCertData

    SSLCertificateFile /etc/pki/tls/certs/sp.pem
    SSLCertificateKeyFile /etc/pki/tls/private/sp.key
    SSLCertificateChainFile /etc/pki/tls/certs/chain.pem
    ...
</VirtualHost>
```

shibboleth2.xml

- Request Map with XML Access control (only option under IIS)
- Applications vs. ApplicationOverride
  - SP's entityID
  - REMOTE-USER preference sequence
  - Session Information (Timeouts, Protocol Handler, Service Locations und Bindings, Session Initiators → federation DS or default IdP, Logout Initiator)
- Credentials
- Attribute Map and Policy
- Metadata Provider (local files or remote URL)
- See https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPShibbolethXML

DARIAH-EU

DAASI International

Filter Attributes in the SP

- attribute-policy.xml
- s. https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPAttributeFilter
- Examples:

```
<afp:AttributeRule attributeID="sn">
    <afp:PermitValueRule
  xsi:type="AttributeIssuerString"
  value="https://testidp.example.org/idp/shib
  boleth"/>
</afp:AttributeRule>
<afp:AttributeRule attributeID="entitlement">
    <afp:PermitValueRule
      xsi:type="AttributeValueString"
    value="urn:mace:dir:entitlement:common-
      lib-terms" />
</afp:AttributeRule>
```

Map Attributes

- attribute-map.xml
- Environment variable `REMOTE_USER` receives special treatment in shibboleth2.xml and not here
- Mapping rules examples:

```
<Attribute name="urn:mace:dir:attribute-
  def:eduPersonScopedAffiliation"
  id="affiliation">
      <AttributeDecoder caseSensitive="false"
        xsi:type="ScopedAttributeDecoder"/>
</Attribute>
<Attribute name="urn:oid:2.5.4.20"
  id="telephoneNumber" aliases="telnr tel"/>
```

SimpleAggregation

- Can request attributes from another IdP than the one the user actually logged in to
- Aggregates attributes through SAML 2.0 attribute queries (via SOAP call to the AA) using an identifier derived from the attributes obtained

```
<!-- use eduPersonPrincipalName, ask for epEntitlement -->
<AttributeResolver  type="SimpleAggregation"
                    attributeId="eppn"
                    format="urn:oid:...1.1.6">
    <Entity>https://ieee.org/idp/shibboleth</Entity>
    <saml2:Attribute
       xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
       Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.7"
       NameFormat="...uri"
       FriendlyName="eduPersonEntitlement"/>
</AttributeResolver>
```

DARIAH-EU

DAASI International

Attribute Checker

- Shibboleth SP >= version 2.5
- Validates a user's session against a list of required attributes (and optionally values) and
  - Either returns the user to complete the login process
  - Or displays an error template (session data available)

```
<Handler type="AttributeChecker"
         Location="/AttrChecker"
         template="attrChecker.html"
         attributes="eppn displayName"
         flushSession="true"/>
```

- Designed to complement the sessionHook:
  - Location where user is sent after session creation
  - HTTP Redirect, params `target` and `return`
  - Hook must either redirect back to SP or lead elsewhere

DARIAH-EU

DAASI International

SP Deployment

- Create server certificates
- Install the SP on the server where the application is
- Operating System Choices:
  - RPM based: download repository file from http://download.opensuse.org/repositories/security:/shibboleth/
  - Debian based/Ubuntu: use shiny new SWITCH repository for SP v2.5, see https://www.switch.ch/aai/docs/shibboleth/SWITCH/2.5/sp/deployment/
  - Mac, Windows, Solaris: see https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSP{Mac|Windows|Solaris}Install

DARIAH-EU

DAASI International

SP (2)

- Configure Apache
  - Vhost for the application
  - Apache access rules
- Configure the SP
  - shibboleth2.xml
  - attribute-map.xml
  - (attribute-policy.xml)
  - IdP Metadata
  - (Certificates)
- Protect something (default: <DocumentRoot>/secure/)
  - Some Test Application: e.g. phpinfo()
  - Your Web application

Troubleshooting

- Log files (CentOS, again :-)
  - /var/log/shibboleth/shibd.log
  - /var/log/shibboleth/transaction.log (z.B. level INFO: IdP, IdP-NameIdentifier, IP address, time, SP sessionID, name of accepted attributes (no value)
  - /var/log/httpd/ssl_access_log
  - /var/log/httpd/ssl_error_log
- Control attributes supplied by the SP using a phpinfo() under /secure/index.php, or
- `https://sp.example.org/Shibboleth.sso/Session`

IdP and SPs

Metadata

- Metadata must be exchanged: IdP needs SP's, and SP needs IdP's metadata
- Shibboleth SP and IdP have an endpoint for metadata retrieval
- Handler addresses:
  - https://exampleSP.org/Shibboleth.sso/Metadata (always generated with all important info)
  - https://exampleIdP.org/idp/profile/Metadata/SAML (IdP maintainer must keep them current)
- Usually a federation amends the generated information and manages metadata distribution

DARIAH-EU

DAASI International

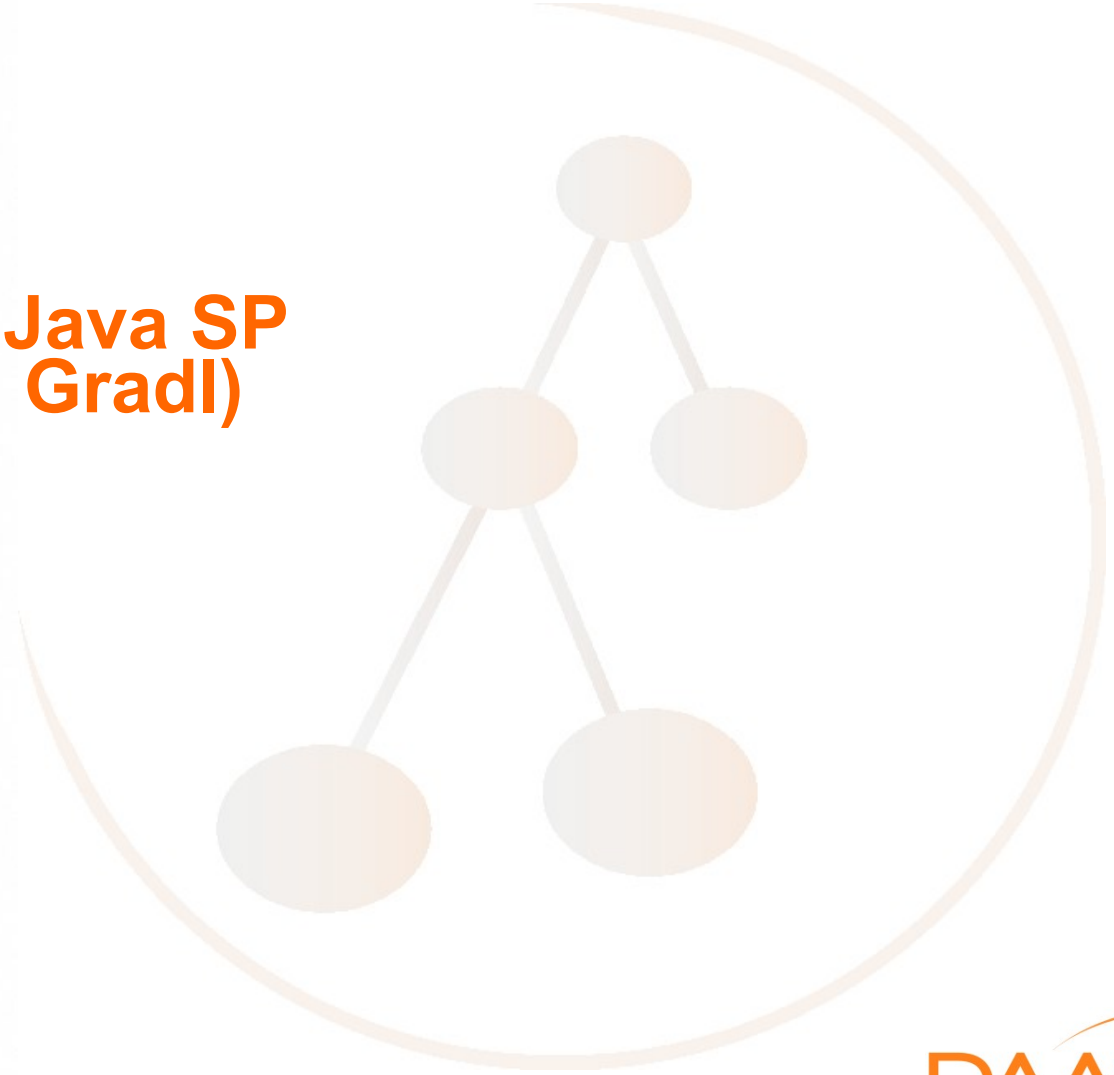
Attribute Exchange

- Compatibility in a federation: IdPs and SPs need to
  - speak the same language (attribute definitions)
  - and must match what they process (attribute exchange policies)
- Thus the following must match
  - attribute-resolver.xml and attribute-map.xml (SP)
  - attribute-filter.xml (IdP) attribute-policy.xml (SP)
- Ideal world!
  - Only in closed environments possible (but there are lots of such)
  - Many parties involved in a federation (there are lots), so it is hard to keep language and policies in synch
  - Many more parties involved with interfederations (eduGain), so even harder

DARIAH-EU

DAASI International

DARIAH Java SP (Tobias Gradl)



DARIAH-EU

DAASI International

Shibbolizing Web Applications

DARIAH-EU

DAASI International

Recommendations for New Web Applications

- Ideally read-only access to user attributes, then these attributes can be provided by environment variables
- Use standard the environment variable $REMOTE_USER
  - Also supported by Tomcat and Apache
  - Can use authentication modules interchangeably
- Use a service account if there needs to be write access to a database (there's not password with SSO)
- If the application has public and closed parts: use different URL paths
- Even easier: protect the whole application with Shibboleth

DARIAH-EU

DAASI International

Shibbolizing Existing Applications

- Points to consider (not a recipe)
  - What is the current authentication method? Apache, Tomcat, LDAP, own mechanism, ...
  - What is the current resource protection method? Apache, Tomcat, own mechanism, ...
  - Does the application have an own session management? How does it work?
  - How does the application's access control work?
  - Any authorization rules for the application – or for those parts that must be protected specifically?
  - User data: From where, What for?
  - Can Shibboleth SP provide these data as attributes?
  - Can IdPs provide them?

(inspired by Bernd Oberknapp, UB Freiburg)

Shibbolizing an Existing Application

- Single Sign-On: the art of removing control of the user password from an application
- Only Web applications
- No authN against a local (at the SP!) data source
  - Application only sees attributes (read only)
  - Any user password remains at the IdP
  - A privileged system account could be used for local read and write data
- *Protect* the Login Path with Shibboleth, and allow for Shibboleth *control* of the rest (to see the attributes)
- Web Server *can* (not required!) allow for access control to the application based on SP attributes

DARIAH-EU

DAASI International

Authentification

- Existing application assumes AuthN by Web server or servlet container?
  - Easy to shibbolize
  - SP Web server module (mod_shib) populates environment / CGI variables
  - Just exchange AuthN method (e.g. Basic Auth, LDAP) in Apache configuration with *shibboleth*
- SP sets special variable REMOTE_USER via a fixed preference list, e.g. „uid eppn persistent-id targeted-id“

Authentification

- More difficult if application
  - Presents its own login screen and/or
  - Assumes to be given login / password
- Usually need a new module that
  - Redirects to the SP Login Handler
  - Saves attributes given by the SP into the application session
- Make that module
  - mandatory
  - or an alternative, but
  - „Dual Log-in“ (login/password plus link to DS on the same page) is not recommended due to phishing awareness

DARIAH-EU

DAASI International

Access Control

- Access control by the Web Server, can be configured in
  - shibboleth2.xml aka *XML access control* (/RequestMap/Host/Path/...)
  - Via Apache directives in httpd.conf or embedded files or via .htaccess files
  - To avoid confusion, should not mix these two mechanisms
  - see https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPProtectContent
  - or https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPContentSettings
- Or application access control (often combined with Lazy Session)

Access Control Pros and Cons

	1.a httpd.conf	1.b .htaccess	2. XML AccessControl	3. Application Access Control
+	▪ Easy to configure ▪ Can also protect locations or virtual files ▪ Regex Support	▪ Dynamic ▪ Easy to configure	▪ Platform independent ▪ Powerful boolean rules ▪ Regex Support ▪ Dynamic	▪ Very flexible and powerful with arbitrarily complex rules ▪ Regex Support
-	▪ Only works for Apache ▪ Not dynamic ▪ Not very flexible rules	▪ Only works for Apache ▪ Only works with “real” files and directories	▪ XML editing ▪ Configuration error can prevent SP from restarting	▪ You have to build it yourself ▪ You have to maintain it yourself

DARIAH

Examples: Apache vs. XML

- Apache Access control in httpd.conf:

```
<Location /secure>
  AuthType shibboleth
  ShibRequestSetting requireSession 1
  Require entitlement common-lib-terms
</Location>
```

- XML-based in shibboleth2.xml:

```
<Path name="secure" authType="shibboleth"
  requireSession="true">
  <AccessControl>
    <Rule require="affiliation">
        member@example.org
    </Rule>
   </AccessControl>
</Path>
```

Application Access Control

- SP only provides for attributes from the IdP
- URL paths must be either under Shibboleth *protection or control*
- Attributes can be used freely by the application for access control, e.g. in a PHP script:

```
if ($_SERVER['affiliation'] == 'staff')
    { grantAccess(); }
```

- E.g. in a Perl CGI script:

```
if ($ENV{'affiliation'} eq 'staff')
    { grant_access() }
```

- E.g. in a Java Servlet in Tomcat proxied by Apache

```
affiliation = (String)
    request.getAttribute("affiliation");
if (affiliation.equals("staff"))
    grantAccess();
```

DARIAH-EU

DAASI International

Lazy Session

- Lazy Session: request Log-in when application needs it
- Usually used with application access control
- Shibboleth must *control* (not *protect*) path:

```
<Location /lazy>
  AuthType shibboleth
  ShibRequestSetting requireSession 0
  Require shibboleth
</Location>
```

- Application requests AuthN by redirecting to:

```
https://sp.example.org/Shibboleth.sso/Login?
    target=https://sp.example.org/cgi-bin/
    application.php&
    entityID=https://idp.example.org/
```

Lazy Session: alternative Login Mechanism

- This variant does not call the LoginHandler directly
- Shibboleth *protect* the URL that usually does Log-in for the application and call it

```
<Location /login>
  AuthType shibboleth
  ShibRequestSetting requireSession 1
  require valid-user
</Location>
```

- At this location, exchange existing password form with
  - reading the user attributes from the SP
  - saving the login information in the session

Example from UB Freiburg

provided by Bernd Oberknapp, aai-users@aai.dfn.de adapted and translated

- Shibbolize a Nagios Server using Basic Auth and local Apache accounts
- (1) Renamed Apache accounts (using the uid from IdP)
- (2) Defined an Entitlement in the IDM for those users that are allowed to access Nagios (alternative: an LDAP group)
- (3) Configured IdP such that uid and Entitlement are released to the SP on the Nagios server
- (4) Configured the SP to use the IdP, accept uid, and provide it as REMOTE_USER
- (5) Include the SP's metadata into the IdP's local metadata

Example from UB Freiburg

provided by Bernd Oberknapp, aai-users@aai.dfn.de adapted and translated

- (6) Change Apache config from LDAP to shibboleth:

```
AuthType Shibboleth
   ShibRequestSetting requireSession 1
   ShibRequireAll On
   Require user ~ ^.+$
   Require entitlement https://mylogin.uni-
                       freiburg.de/entitle
                       ment/ub/nagios/admin
```

- What the particular user is allowed to see (authZ) remains in the Nagios configuration
- Accordingly, Nagios configuration must be touched if such user leaves the university

Advanced Techniques

- isPassive at the Service Provider
  - Only for SAML2 SPs
  - Allows for Log-in to the application without any user interaction
  - Application must be protected with Lazy Session
- IdP LoginHandler must support it
  - UsernamePassword Handler does
  - RemoteUser Handler does not
- Check via redirect to the IdP, without the user noticing
  - If there's no IdP session, does not present IdP login screen but come back, sending to some default page
  - If there's a session, user comes back authenticated

DARIAH-EU

DAASI International

Advanced Techniques

- forceAuthn mechanism
  - SAML2
  - Request renewed authN for critical operations at an application
  - Precedes over SSO
  - Must be supported by the IdP (Shibboleth IdP does)

SP Deployment Options

- Full Shibboleth protections or Lazy Session?
- Access control by container or by application?
- Personalization via
  - pseudonyms
  - REMOTE-USER
  - Further user attributes
- One IdP, a few IdPs, or an embedded or central Discovery Service?
- Which federation(s)?
- One logical SP (~vhost) or many?
- Further authN methods besides Shibboleth?

CLARIN and Shibboleth: Integration into the LAT software stack (Willem Elbers)

DARIAH-EU

DAASI International

JEE Strategies

DARIAH-EU

DAASI International

Java Applications

- Shibboleth SP is a Web server module
- But J2EE applications run in a Servlet Container (Tomcat, Jetty, ...) or Application Server (JBoss, Geronimo, ...)
- Solution: Web Server Proxy (e.g. mod_proxy_ajp)
- Other Options
  - DARIAH Java SP!!!
  - Spring Security Shibboleth Native SP plugin for Grails/Groovy/Java Web applications
  - Non-Shibboleth SAML SP products, e.g. OpenAM, ESOE, OIOSAML.java, (s. http://saml.xml.org/wiki/saml-open-source-implementations)

Java Applications (2)

- Role concept: the container...
  - sets the User Principal (REMOTE_USER)
  - Assigns principals to Roles, declaratively
- Container Managed Security (web.xml)
  - Translate <security-constraint>s and <web-resource-collection>s into Apache <Location>s
  - Omit <login-config>s und <security-config>s
  - Not recommended for complex roles
- Application Security (servlet filter)
  - request.getRemoteUser() → can be used directly
  - request.isUserInRole(role) → proxy with a HttpServletWrapper
  - Lazy Session: access control per Servlet

Application Managed Security: web.xml

```
<web-app>
  <display-name>Attribute2Role</display-name>
  <filter>
   <filter-name>Attribute2RoleFilter</filter-name>
   <filter-class>de.sc.Attribute2RoleFilter</filter-class>
  </filter>

  <servlet>
    <servlet-name>ListParameters</servlet-name>
    <servlet-class>de.sc.ListParameters</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>ListParameters</servlet-name>
    <url-pattern>/ListParameters</url-pattern>
  </servlet-mapping>

  <filter-mapping>
    <filter-name>Attribute2RoleFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

Application Managed Security: Filter

```
public class Attribute2RoleFilter implements Filter  {

    @Override
    public void doFilter(  ServletRequest request,
                           ServletResponse response,
                           FilterChain chain) {
        ServletRequest req = new HttpRequestWrapper(
                               (HttpServletRequest) request);
        chain.doFilter(req, response);
    }
}
```

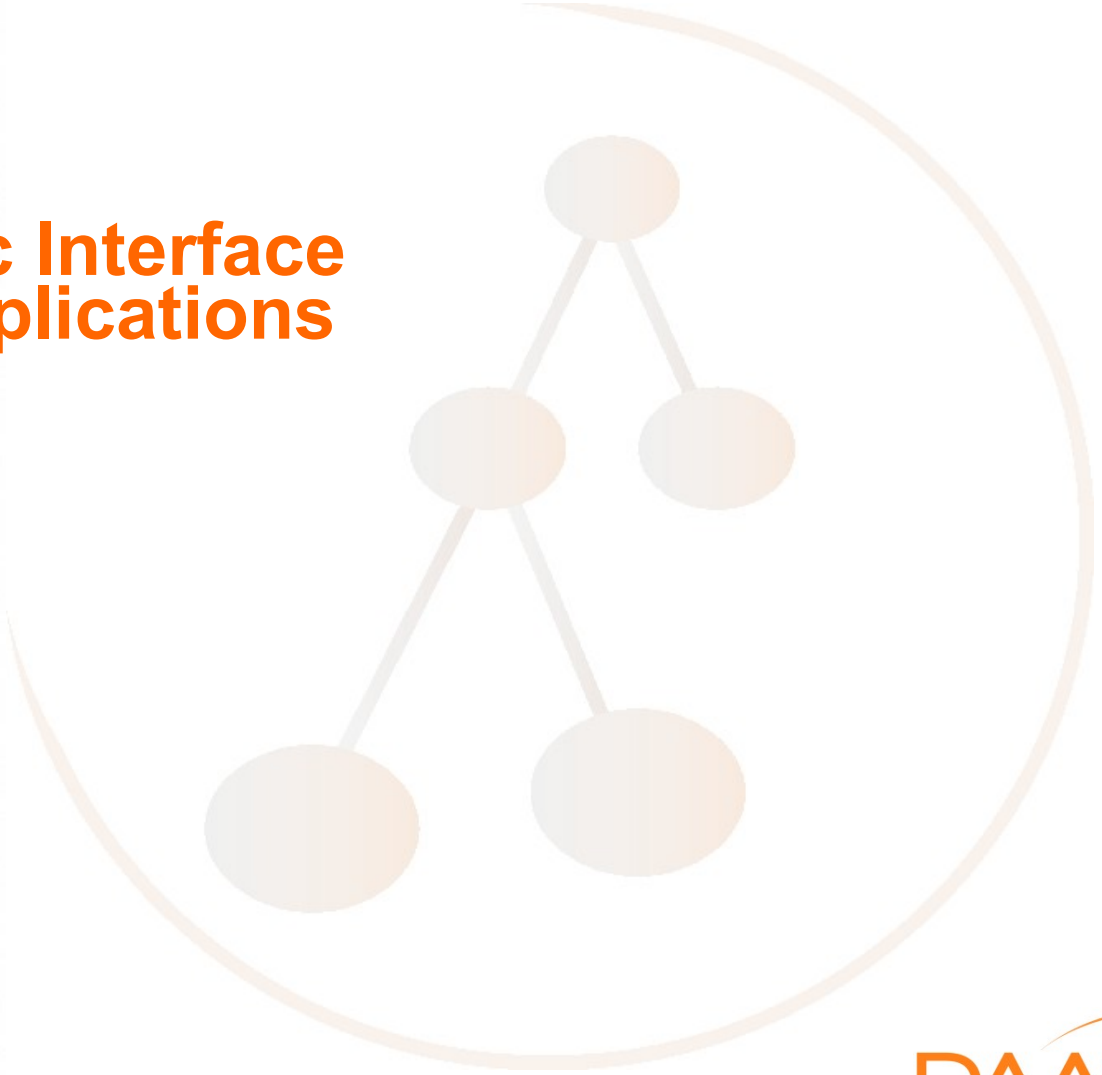
Application Managed Security: Wrapper

```
public class HttpRequestWrapper extends
                                    HttpServletRequestWrapper {
    private String surname;
    private boolean loser;

    public HttpRequestWrapper(HttpServletRequest request) {
        super(request);
        surname = (String) request.getAttribute("sn");
        String attr = (String) request.getAttribute("loser");
        if ((attr != null) && attr.equals("TRUE"))
                loser = true;
        else   loser = false;
    }
    @Override
    public boolean isUserInRole(String arg0) {
        if (arg0.equals("loser"))
            return loser;
        return false;
    }
    public String getSurname() {
        return surname;
    }
}
```

Application Managed Security: Servlet

```
public class ListParameters extends HttpServlet {

    protected void doGet(...)  // call doIt
    protected void doPost(...) // call doIt

    void doIt(HttpServletRequest request,
              HttpServletResponse response) {
    out.println("<html>");

    //...

    out.println("Surname (sn): "+ request.getAttribute("sn");

    if (request.isUserInRole("loser")) out.println("Loser!");
       else out.println("Winner!");

    if (request instanceof HttpRequestWrapper) {
       String surname = ((HttpRequestWrapper) request)
                              .getSurname();
       out.println("cached value): " + surname);
       }
    }
}
```

Generic Interface for Applications



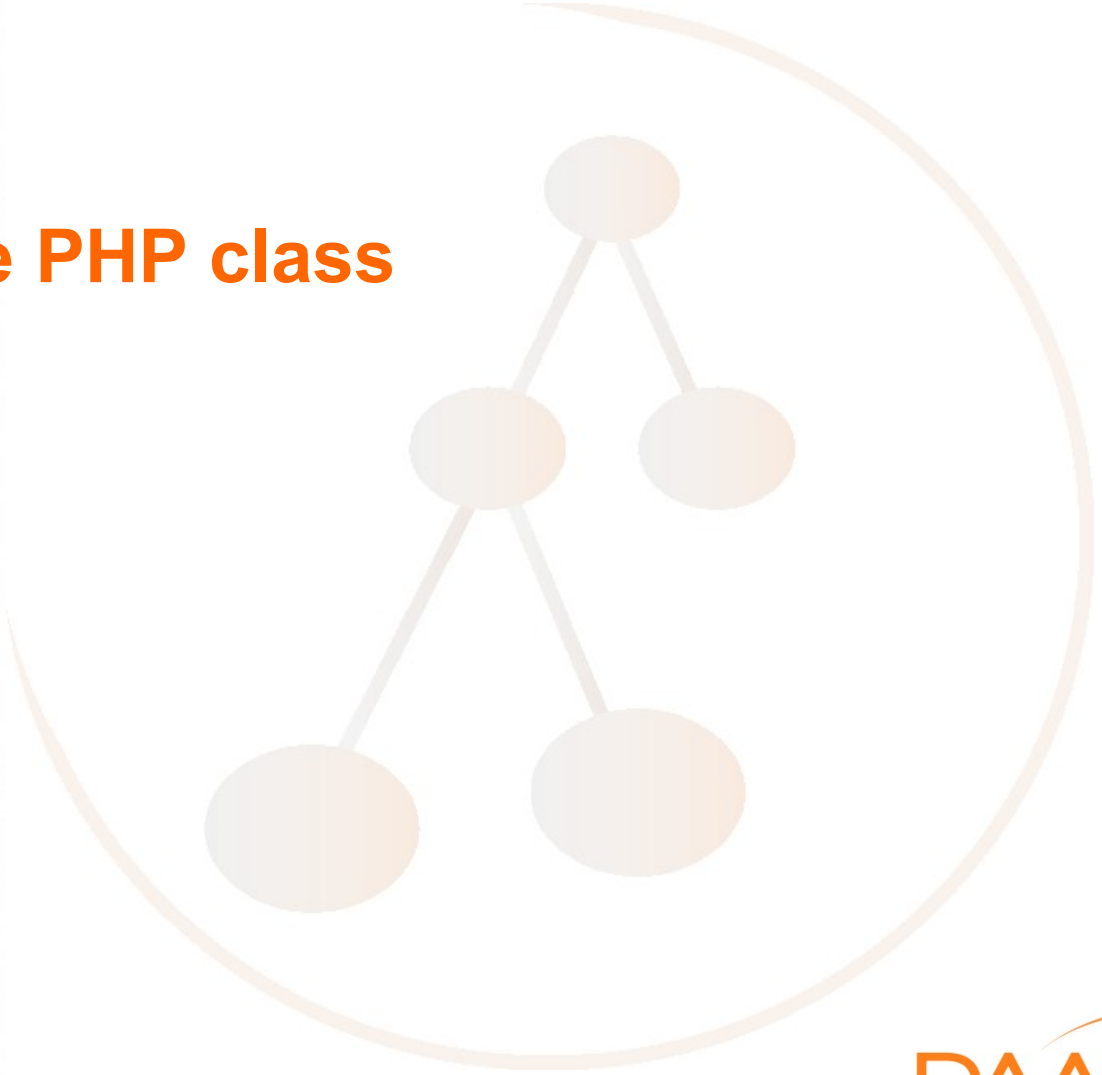
DARIAH-EU

DAASI International

Interface

- In order to switch to SSO easily, an application should request the following methods:
  - `Boolean is_authenticated ()`
  - `void authenticate ( URL target, String entityID )`
    - or alternatively: `URL get_authentication_URL ( URL target, String entityID )`
  - `void logout ( URL return )`
    - or alt. `URL get_logout_URL ( URL return )`
  - `String get_user_id ()`
  - `Array [String] get_attribute ( String attribute_name )`
  - `Array [ Array [String] ] get_attributes ()`
- Other SSO frameworks have a similar API, e.g. CAS, OpenID etc.

Example PHP class



Interface

```
interface INTERFACE_AUTH
{
  public function is_authenticated ();
  public function authenticate (
     $target_URL, $idp_entity_id );
  public function get_authentication_URL (
     $target_URL, $idp_entity_id );
  public function logout ( $return_URL );
  public function get_logout_URL(
     $return_URL ) ;
  public function get_user_id ();
  public function get_attribute ( $name );
  public function get_attributes ();
}
```

Initialization

```
class SHIB_AUTH implements INTERFACE_AUTH
{
  private $default_shib_login_handler =
     "/Shibboleth.sso/Login";

  private $default_shib_slo_handler =
     "/Shibboleth.sso/Logout";

  private $default_shib_attributes =
     array ( "sn",
             "givenName",
             "cn",
             "mail",
             "eppn" );
```

Login Status

```
public function is_authenticated ()
   {
     if (isset ( $_SERVER['Shib-Session-ID']))
        {
    return TRUE;
        }
      else
        {
    return FALSE;
        }
   }
```

Login

```
public function authenticate ( $target_URL = NULL,
                                $idp_entity_id = NULL )
  {
    header( 'refresh:0;url=' . $this->get_authentica-
            tion_URL ( $target_URL, $idp_entity_id) );
    flush();
    exit;
  }

public function get_authentication_URL (
             $target_URL = NULL, $idp_entity_id = NULL )
  {
    return $this->shib_login_handler .
      ( $target_URL == NULL ? '' : '?target=' .
$target_URL ) .
      ( $idp_entity_id == NULL ? '' : '&entityID=' .
$idp_entity_id );
  }
```

Logout

```
public function logout ( $return_URL )
   {
     header( 'refresh:0;url=' .
          $this->get_logout_URL ( $return_URL) );
     flush();
     exit;
   }

public function get_logout_URL ( $return_URL )
   {
     return $this->shib_slo_handler .
       ( $return_URL == NULL ? '' : '?return=' .
$return_URL );
   }
```

Attributes

```
public function get_user_id () {
    return $_SERVER['REMOTE_USER'];
  }

public function get_attribute ( $name )
  {
    return explode ( ';', $_SERVER[$name] );
  }

public function get_attributes ()
  {
    $result = array();
    foreach ($this->shib_attributes as $a)
      {
      $result[$a] = $this->get_attribute ( $a );
      }
    return $result;
  }
```

Use the Class

```
$myUrl = "https://example.org/index.php
include_once ("class.shib_auth.php");
$shibsession = new SHIB_AUTH("shib_conf.php");

if ( isset( $_POST['doLogout']))
  {
    $shibsession->logout( $myUrl . "?
LogoutHint" );
  }
elseif ( isset( $_POST['doLogin']))
  {
    $shibsession->authenticate( $myUrl );
  }
```

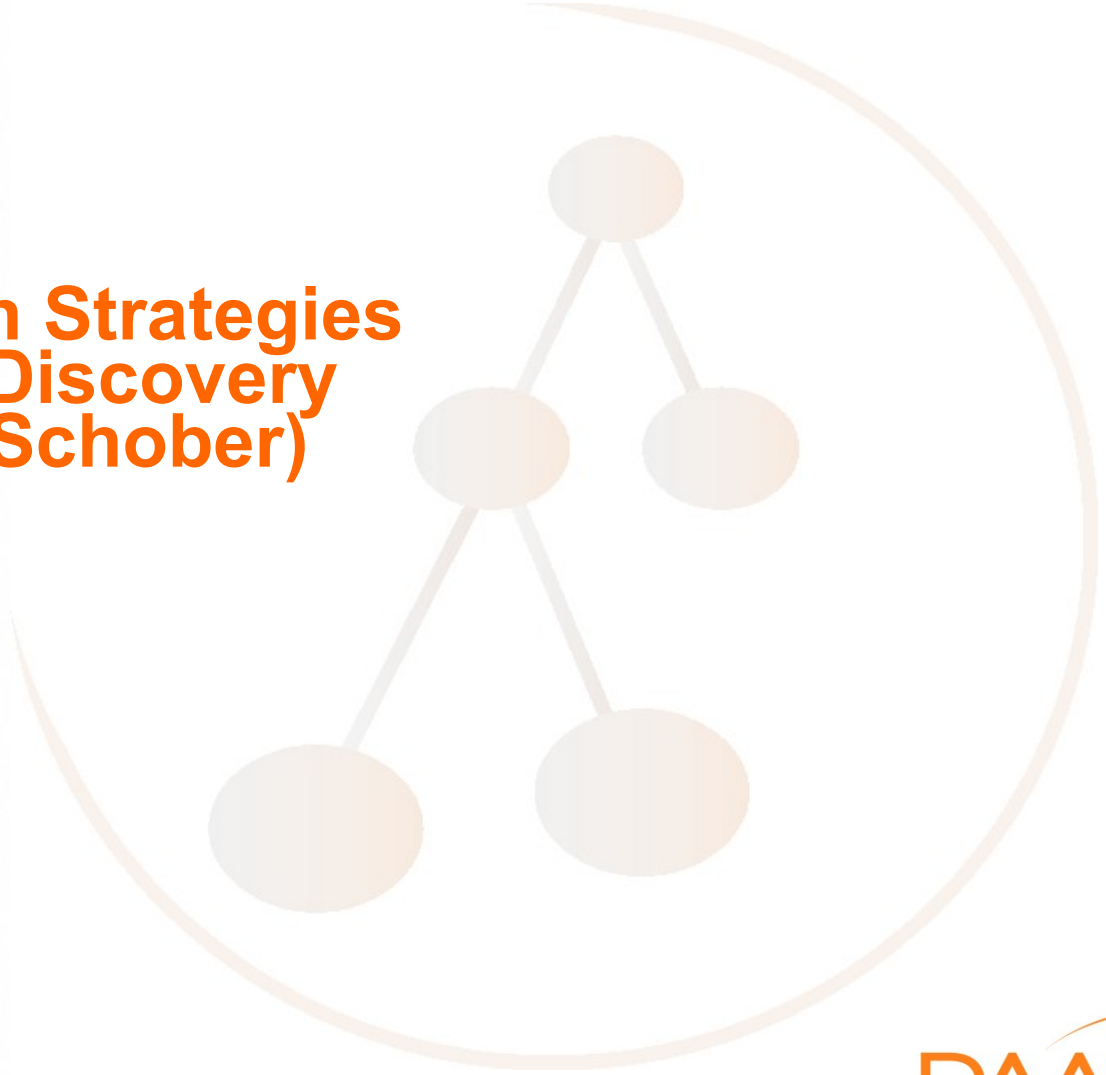
DARIAH-EU

DAASI International

Use the Class

```
elseif ($shibsession->is_authenticated())
   {
    // echo „valid session exists“
    // dump user ID and attributes
    // echo form with logout button
    //      using hidden doLogout field
   }
else
   {
    // give a SSO-logout hint if after logout
    // echo „valid session does not exist“
    // echo form with login button
    //      using hidden doLogin field
   }
```

Integration Strategies for IdP Discovery (Peter Schober)



DARIAH-EU

DAASI International

User Experience

- Some SPs (e.g publishers) do not need personal data, a pseudonym suffices for personalization
- However, many SPs need personal data (e.g. in Research Education) to operate correctly
- Usually, a federation provides for common metadata, but does not rule attribute release
- Attribute release is subject to legal restrictions, i.e. privacy and data protection
- One solution: be liberal in releasing attributes, but let the user decide → IdP Extension *uApprove*
- Another solution: make bilateral arrangements. This influences IdP choice for a service...

User Experience

Inspired by Peter Schober

- Various possible IdP Discovery strategies for an SP
- A: Central DS of the federation
  - Limited to IdPs in one federation
- B: Embedded DS, filled from that SP's Metadata
  - All possible IdPs
- C: Embedded DS with IdP whitelist
  - Only a set of specific IdPs available
- From a user experience point of view, A and B can lead to unwanted results:
  - User chooses home IdP and logs in successfully
  - However, that IdP might not „know“ the SP
  - Thus no required attributes released
  - Thus authorization error

DARIAH-EU

DAASI International

User Experience

Inspired by Peter Schober

- Approaches A + B show all IdPs
  - Need special error treatment in the application
  - However, this is the chance to change something:
    - Display IdP name and logo,
    - Tell the concrete reason: which attribute is missing
    - Display IdP contact details to request for attribute exchange with this SP
  - → User can ask at the right place, i.e. at the IdP
- Approach C shows only 'known good' IdPs
  - IdP list is harder to maintain
  - Less disappointing for users
  - But: If „my“ IdP is not in the list, there's no easy way for me to request this (as the SP is the wrong address)

DARIAH-EU

DAASI International

User Experience

- New construct: Code of Conduct (CoC)
  - SPs assert that they adhere to privacy protection
  - SPs describe what personal data they process
- Technical implementation: SAML entity attributes
  - SAML metadata extension
  - http://www.geant.net/uri/dataprotection-code-of-conduct/v1
  - Give PrivacyStatementURL
  - List required and optional RequestedAttributes
- IdPs can base attribute release policy upon these entity attributes → bilateral arrangements are not required anymore, IdP maintenance is easier

DARIAH Authorization Framework

DARIAH-EU

DAASI International

Authorization

- Many frameworks and standards exist for Authentication (= logging users in): Basic AuthN, LDAP, SAML, X.509, ...
- Only few standards for Authorization (=granting access to some resource): XACML, RBAC and ...?
- Two possible ways to do the actual authorization decision:
  - Every service / application using own set of rules
  - Centrally managed service containing access rules and all services must ask this service for access decision
- Proposed a mixture of both possibilities
  - Basic authorization is stored centrally (access groups in DARIAH LDAP
  - However, it is upon the service to honor these attributes and to authorize the user upon their presence and/or content and other attributes

DARIAH-EU

DAASI International

Motivation

- Challenges:
  - CoC has not arrived fully in today's federations yet
  - DARIAH SPs need some personal data: mail, surname, givenname, organization, eduPersonPrincipalName
  - Users' Home („Campus“) IdPs do not provide these
  - Service Authorization: DARIAH central LDAP directory maintains list of user groups that regulate access to e.g. the DARIAH Wiki
  - Aim: also allow services protected by SAML SPs to base their Authorization decisions upon such group membership
- Developed AAI concept satisfying these issues
- We're not the first ones: see
  http://www.switch.ch/aai/downloads/20090908-JRA3-SAML-VO-Platform.pdf

DARIAH-EU

DAASI International

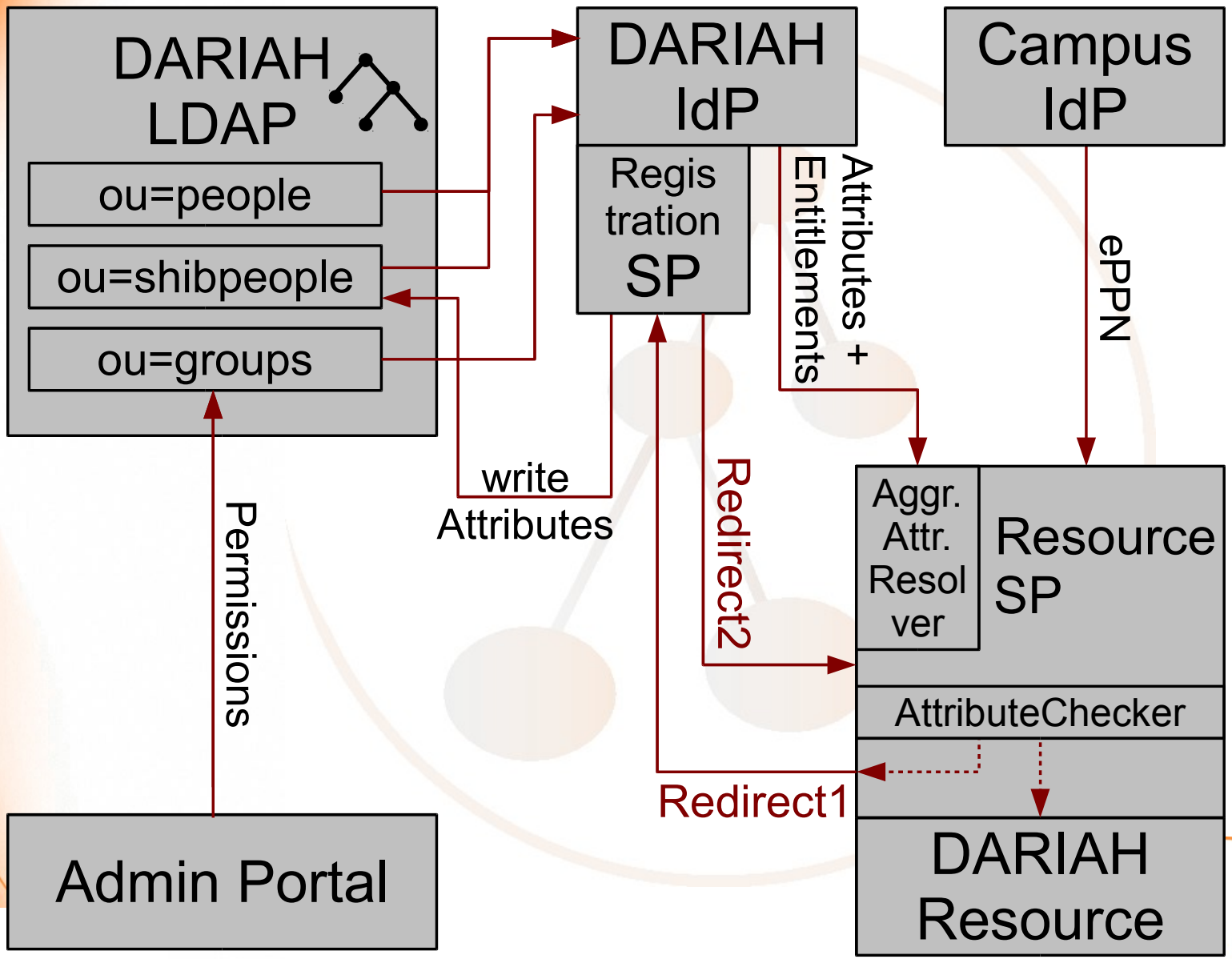
Introducing DARIAH SP Blueprint

- „Blueprint“: solution for each SP in the DARIAH AAI that allows for authentication by Campus IdPs
- Basic idea:
  - User tries to access DARIAH SP
  - Authentication at Campus IdP (front channel)
  - SAML Attribute Request at the DARIAH IdP (back channel without user interaction), providing:
    - Attributes not given by Campus IdP
    - Group memberships
- There is one „registration“ page upon first time access
  - Protected by an SP
  - Complementing required attributes
  - Approval of DARIAH Terms of Use

DARIAH-EU

DAASI International

DARIAH AAI Infrastructure



Premises

- Usage of a federated AAI, e.g. DFN-AAI or eduGAIN
- Account at either Campus IdP or DARIAH homeless IdP
- If Campus IdP: need at least one Identifier (ePPN) at the SP
- DARIAH LDAP, containing
  - Homeless users branch (attributes and passwords)
  - Federated users branch (only attributes, provided by the Campus IdP and/or by the user)
  - Authorization groups branch, with members referencing the other two branches
- A resource / DARIAH service, protected by a Shibboleth SP which takes part in the federation
- Management of authorization groups by the Admin Portal

DARIAH-EU

DAASI International

AAI Flow (1)

- User wants to access some resource protected by an SP
- Resource SP sends the user to their Campus IdP for authentication and creates a session upon return
- Resource SP resolves user attributes from both the
  - Campus IdP: eduPersonPrincipalName
  - DARIAH IdP: eduPersonEntitlement based on the user's group membership, and other personal data
- The received attributes are checked
  - Enough attributes? Grant access to resource
  - Not enough attributes? Continue on next slide.

DARIAH-EU

DAASI International

AAI Flow (2)

- Delete Session at SP and redirect to registration web form
  - Form ist protected by an SP as well
  - Form receives HTTP GET attributes
    - User's Campus IdP EntityID, for automatic log-in (SSO)
    - URL of the resource the user initially wanted to access
  - Attributes provided by the Campus IdP are presented read-only
  - Input fields for missing attributes are provided
  - Checkbox for the DARIAH Terms Of Use
- On form submission, these details are entered into the DARIAH LDAP branch for federated users
- HTTP Redirect back to the requested resource. Continue on previous slide.

Limitations

- Works with ECP, but missing attributes are filled in interactively, so need WebSSO profile for first time access
- Campus IdP should release same set of attributes to a DARIAH Resource SP as to the Registration SP
- DARIAH LDAP knows federated user IDs only after their first login
  - Adding users via Admin portal to groups only possible after registration
  - But: there are other ways to pre-populate groups in LDAP
- Not yet a usable concept for Attribute *updates* at the Campus IdP and their inclusion into the DARIAH LDAP once registration was finished

Blueprint: Implementation

in /etc/shibboleth/shibboleth2.xml:

```
<ApplicationDefaults ...targeted-id"
  sessionHook="/Shibboleth.sso/AttrChecker">

...
      <Handler type="AttributeChecker"
               Location="/AttrChecker"
               template="attrChecker.html"
               attributes="eppn mail givenName"
               flushSession="true"/>

      <AttributeResolver type="SimpleAggregation"
            attributeId="eppn"
            format="urn:oid:...1.1.1.6">
         <Entity><DARIAH-IDP></Entity>
      </AttributeResolver>
```

Blueprint: Implementation

in /etc/shibboleth/attrChecker.html:

```
<html><head>
   <meta http-equiv="refresh" content="3;
      URL=https://ldap-dariah.esc.rzg.mpg.de/
          Shibboleth.sso/Login
            ?target=
              /secure/UserAttributesCompletion.php
                 %3ForiginalURL%3D<shibmlp target/>
            &entityID=<shibmlp entityID/>
         "/>
    <title>Insufficient Information</title>
</head><body>
   <h2>Insufficient Information</h2>
   Your home organisation did not provide
   sufficient attributes to this service.
   Therefore, you are now being redirected to the
DARIAH central user registry. Details:...
</body>
</html>
```

Blueprint: Implementation

in /etc/shibboleth/attribute-policy.html:

```
<!-- deny attributes, except from Dariah IdP -->
<afp:AttributeFilterPolicy>
   <afp:PolicyRequirementRule xsi:type="NOT">
      <Rule xsi:type="basic:AttributeIssuerString"
            value="<DARIAH-IDP>" />
   </afp:PolicyRequirementRule>

   <afp:AttributeRule attributeID="*">
      <afp:DenyValueRule xsi:type="ANY"/>
   </afp:AttributeRule>
</afp:AttributeFilterPolicy>
```

We need this only because the Shibboleth SP cannot de-duplicate attributes with the same values...

DARIAH-EU

DAASI International

Blueprint: Implementation

in /etc/shibboleth/attribute-map.html:

```
<Attribute
   name="urn:oid:1.3.6.1.4.1.5923.1.5.1.1"
   id="isMemberOf" />
```

..that's all!

Thank You!

- Questions?

- DAASI International
  - www.daasi.de
  - Info@daasi.de