Track D: Securing Online Services in the DARIAH AAI using SAML/Shibboleth

DESIR Code Sprint
Berlin

Speaker: Martin Haase
DAASI International

Date: 31 Jul – 02 Aug 2018

Agenda – Theory Bits

1. Welcome and Introduction to Track D – DARIAH AAI
2. Single Sign-On (SSO) concepts for Web applications - IdPs and SPs
3. Federations and eduGAIN
4. Introduction to SSO using SAML
5. Introduction to Shibboleth as a language-independent SP solution
6. On top of SAML and eduGAIN: benefits of the DARIAH AAI
7. Architecture of the DARIAH AAI IdP Proxy Solution
8. Other SP implementations
9. Other SSO technologies (OAuth2, OpenID Connect)

Agenda – Hands-On

1. Install the Shibboleth SP on Linux
2. Configuration against the DARIAH AAI Proxy IdP
3. Vhosting / using logical SPs
4. Initiating a session (active and passive protection)
5. Authorization options: application-based, htaccess, XML-based
6. Interaction with applications in various programming languages
7. Using DARIAH AAI attributes for authorization
8. Management of authorization groups using the DARIAH SelfService

Single Sign-On Concepts

Components

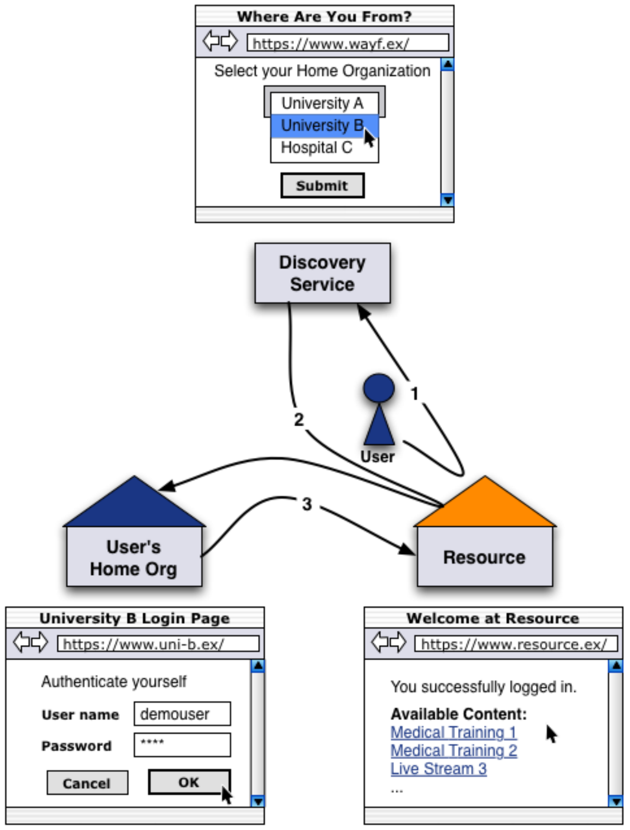
- Identity Provider (IdP)
  - Central component for authentication (Log-In)
  - Provides user attributes (Attribute Authority, AA)
- Service Provider (SP)
  - Protects Web resources
  - Provides applications with user attributes
- Discovery Service (DS)
  - Lets users choose their IdP

Terms

- Federation
  - Crosses organizational boundaries
  - Union of many SPs and IdPs
  - Central provider of mutual trust SPs ↔ IdPs
- Single Log-On
  - Use the same credentials for different services
- Single Sign-On (SSO)
  - Only one log-on event per session, or day

Short introduction to Web SSO...

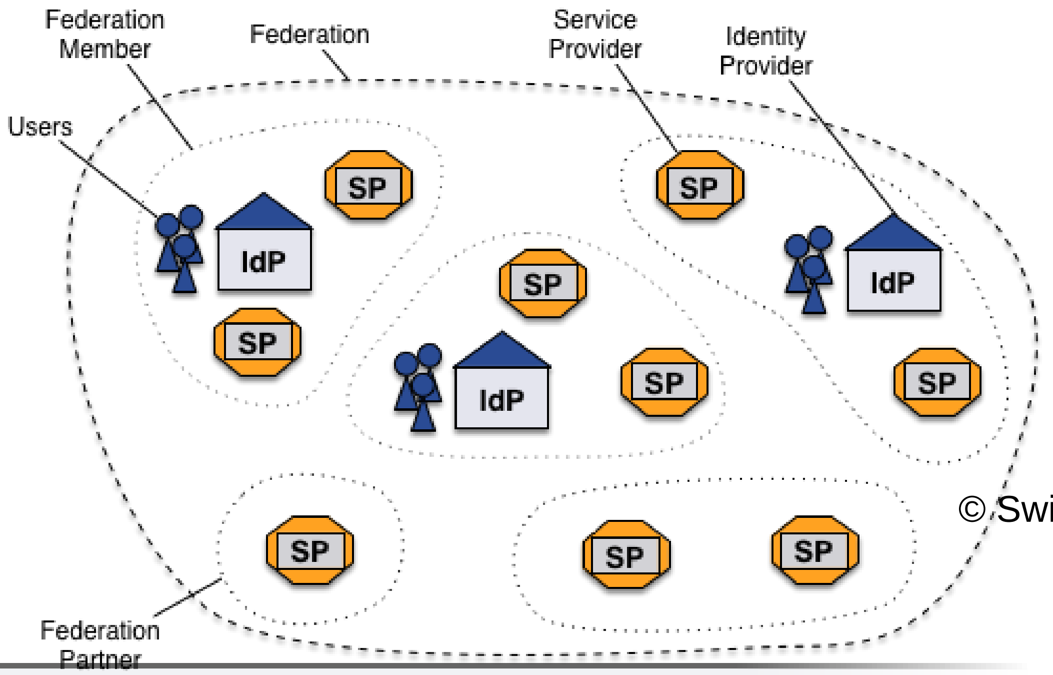
- Most important use case: Web Browser SSO
- Identity Provider (IdP) authenticates users and issues assertions
- Service Provider (SP) relies on assertions from IdP and uses this information to control access to protected services
- Discovery service (DS) lets users choose their home organization (and therefore IdP)



© Switch.ch

… and federations

- Federations as collection of organizations
- Federation operator as trusted third party
- Scalable way to allow SSO across organizational boundaries
- SAML Metadata to connect entities

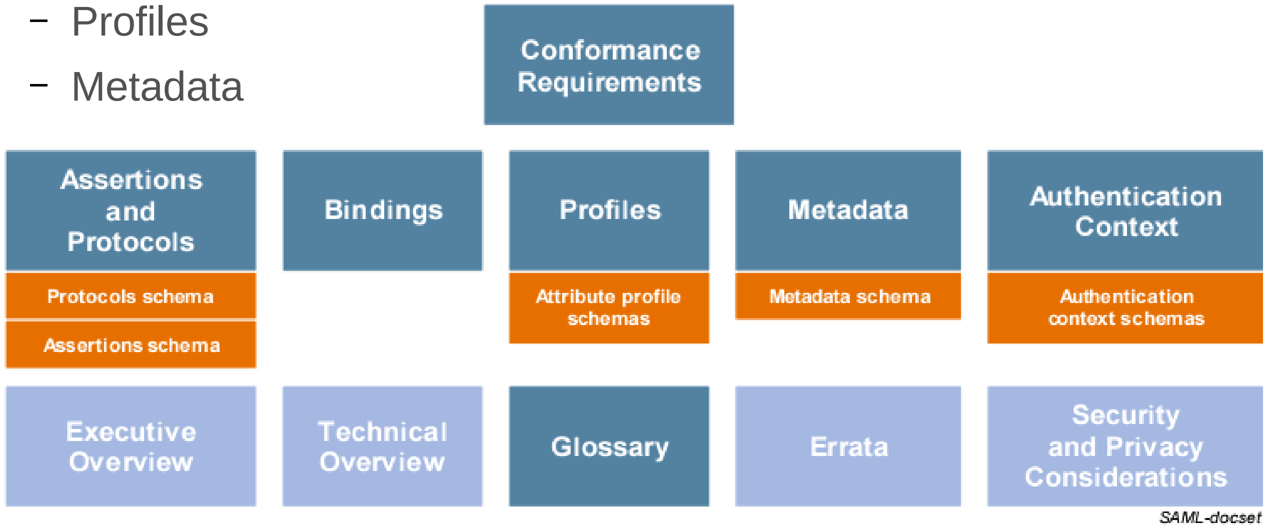


© Switch.ch

SSO using SAML

SAML Basics

- Security Assertion Markup Language (SAML)
- Exchange of informationen between SAML Entities
  - Asserting Party (IdP)
  - Relying Party (SP)
- OASIS standard, current version 2.0, March 2005
- Main components:
  - SAML Core: Assertions und Protocols
  - Bindings
  - Profiles
  - Metadata



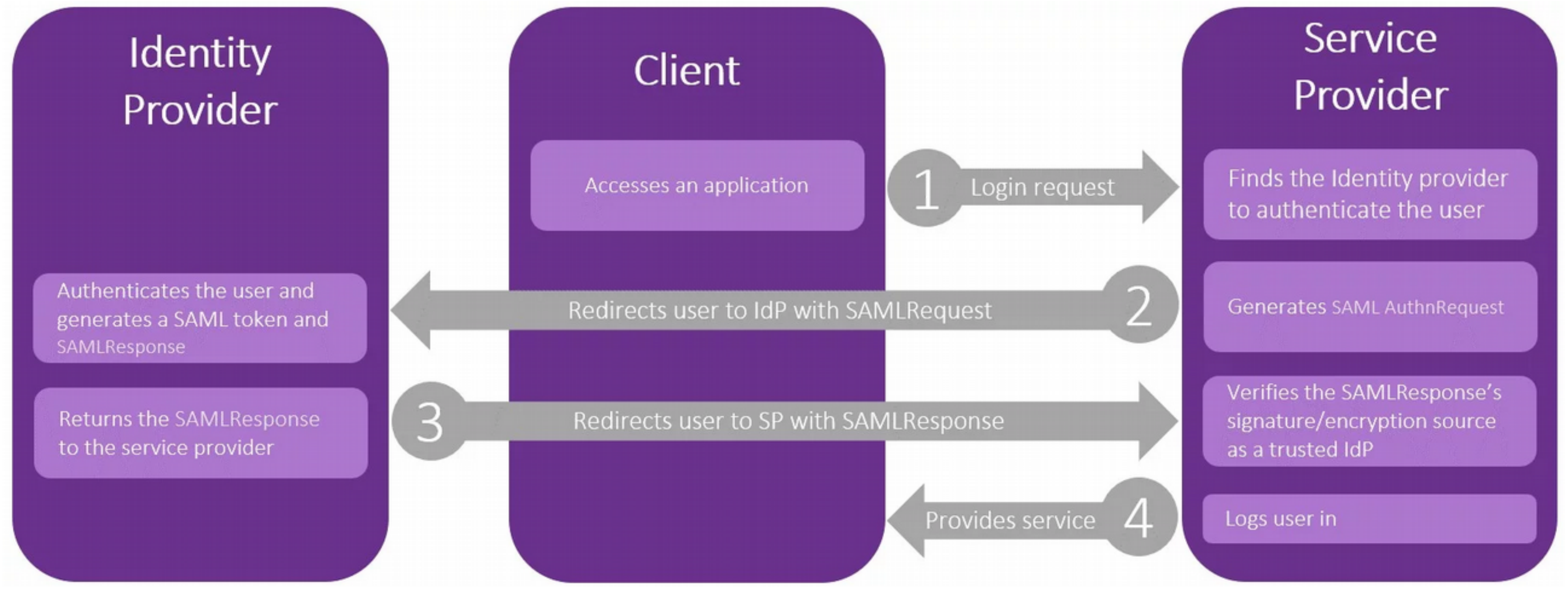
SAML Glossary

- Components: IdP, SP, Discovery Service
- Terms: Federation, SSO
- SAML Entity ID: ID of an IdP or SP (looks like a URL often)
- SAML Assertion: information about a subject, with details about the authentication event, user attributes, …
- Attribute Authority: an IdP endpoint that releases attributes
- Session: „Security context“, both the IdP and the SP keep a session for the user

„Top down“: SAML Profiles

- Description of use cases
  - Definition of SAML messages (→ Assertions und Protocols)
  - Definition of the transport mechanism (→ Bindings)
- SSO Profiles
  - Web Browser SSO Profile
  - Enhanced Client or Proxy (ECP) Profile
- Further Profiles
  - IdP Discovery
  - Single Logout
  - Artifact Resolution
  - Attribute Query
  - Etc.

Web Browser SSO Profile

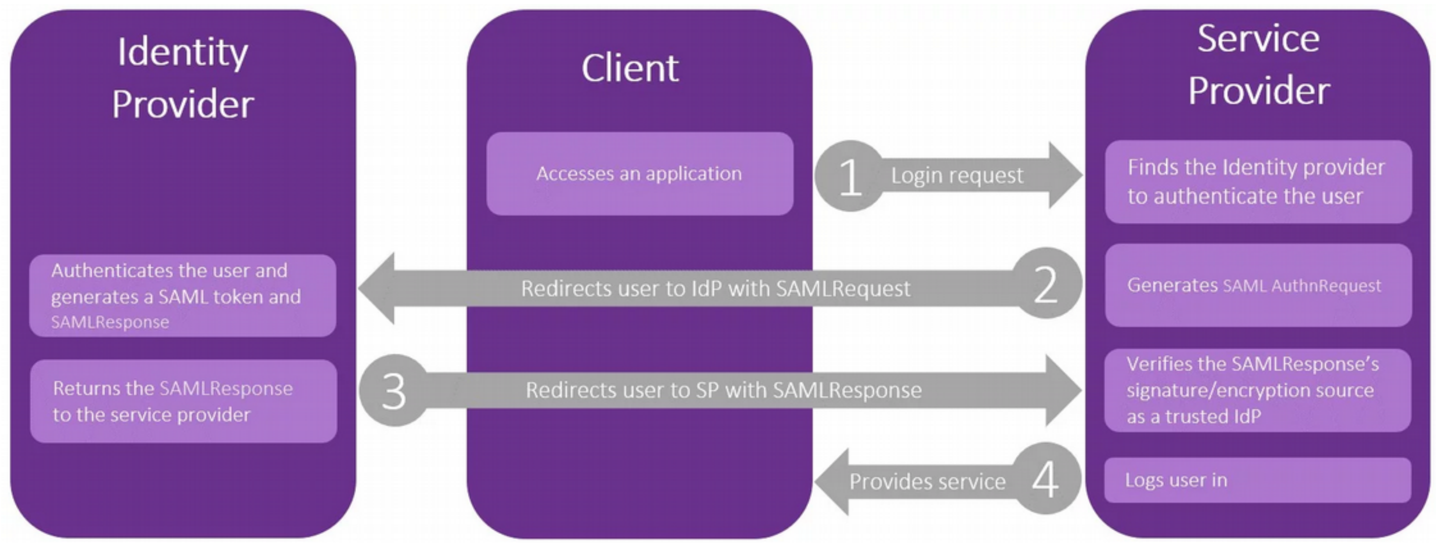

Identity Provider
Client
Service Provider
Accesses an application
1 Login request
Finds the Identity provider to authenticate the user
Authenticates the user and generates a SAML token and SAMLResponse
Redirects user to IdP with SAMLRequest 2
Generates SAML AuthnRequest
Returns the SAMLResponse to the service provider
3 Redirects user to SP with SAMLResponse
Verifies the SAMLResponse's signature/encryption source as a trusted IdP
Provides service 4
Logs user in

© cyberark.com

Protocols

- Specify XML Schema for SAML protocol messages
- Abstract form
  - RequestAbstractType
  - ResponseAbstractType
- Concrete Examples
  - <AuthnRequest> (SP → IdP)
  - <Response> (IdP → SP)
  - <LogoutRequest> (SP → IdP, IdP → SP)
- Specify validation of SAML messages

<AuthnRequest>



Identity Provider
Client
Service Provider
Accesses an application
1 Login request
Finds the Identity provider to authenticate the user
Authenticates the user and generates a SAML token and SAMLResponse
Redirects user to IdP with SAMLRequest 2
Generates SAML AuthnRequest
Returns the SAMLResponse to the service provider
3 Redirects user to SP with SAMLResponse
Verifies the SAMLResponse's signature/encryption source as a trusted IdP
Provides service 4
Logs user in

```
<samlp:AuthnRequest
AssertionConsumerServiceURL="https://sp.org/Shibboleth.sso/SAML2/POST"
Destination="https://idp.org/idp/profile/SAML2/Redirect/SSO"
ID="_926bb3c39ce15feb5f07595badd2e17a"
IssueInstant="2017-11-27T14:39:48Z"
ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Version="2.0" xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
  <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
    https://sp.org/shibboleth
  </saml:Issuer>
  <samlp:NameIDPolicy AllowCreate="1"/>
</samlp:AuthnRequest>
```

<Response>

```
<saml2p:Response Destination="https://sp.org/Shibboleth.sso/SAML2/POST"
ID="_e799fb247f68025b1971a1ab6fa7a5c7"
InResponseTo="_926bb3c39ce15feb5f07595badd2e17a"
IssueInstant="2017-11-27T14:48:52.792Z" Version="2.0"
xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol">
  <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
    https://idp.com/idp/shibboleth
  </saml2:Issuer>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    …
  </ds:Signature>
  <saml2p:Status>...</saml2p:Status>
  <saml2:Assertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
    …
  </saml2:Assertion>
</saml2p:Response>
```

<Assertion>

```
<Assertion>
  <Issuer>...</Issuer>
  <Subject>
    <NameID>UserIdentifier</NameID>
    ...
  </Subject>
  <Conditions>...</Conditions>
  <AuthnStatement>
    <AuthnContext>
      <AuthnContextClassRef>
        urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
      </AuthnContextClassRef>
    </AuthnContext>
  </AuthnStatement>
  <AttributeStatement>
    <Attribute Name="uid">
      <AttributeValue>test</AttributeValue>
    </Attribute>
  </AttributeStatement>
<Assertion>
```

- Usually an <EncryptedAssertion> → XML Encryption

Bindings

- Specify how SAML protocol messages are bound to common transport protocols
- HTTP Redirect Binding
- HTTP POST Binding (→ „Auto Submit Form“)
- HTTP Artifact Binding
- SAML SOAP Binding
- Reverse SOAP Binding
- SAML URI Binding

E.g. HTTP Redirect Binding

```
HTTP/1.1 302 Object Moved
Date: 26 Nov 2017 16:00:49 GMT
Location:
https://idp.org/SAML2/SSO/Redirect?
SAMLRequest=fZFfa8IwFMXfBb9DyXvaJtZ1BqsURRC2Mabbw95i
vc5Am3TJrXPffmmLY3%2FA15Pzuyf33On8XJXBCaxTRmeEhTEJQB
dmr[...]
```

- SAMLRequest=URLEncode(base64(DEFLATE(x)))
- x: SAML protocol message
  - E.g. an <AuthnRequest>

Security Mechanisms in SAML

- SAML Metadata as a base for SP ↔ IdP trust, defined by the SAML-Standard
- SPs und IdPs benötigen need information about each other
  - Description of the communication partner‘s capabilities
  - Endpoint locations (URLs etc.)
  - Inline PKI certificate for XML signature and encryption
- Use OIDs (from LDAP) for unambiguous attribute name encoding, e.g. givenName *urn:oid:2.5.4.42*
- Server clocks must be synchronized, use of TLS, etc

Introduction to Shibboleth

DAASI International

Shibboleth Project

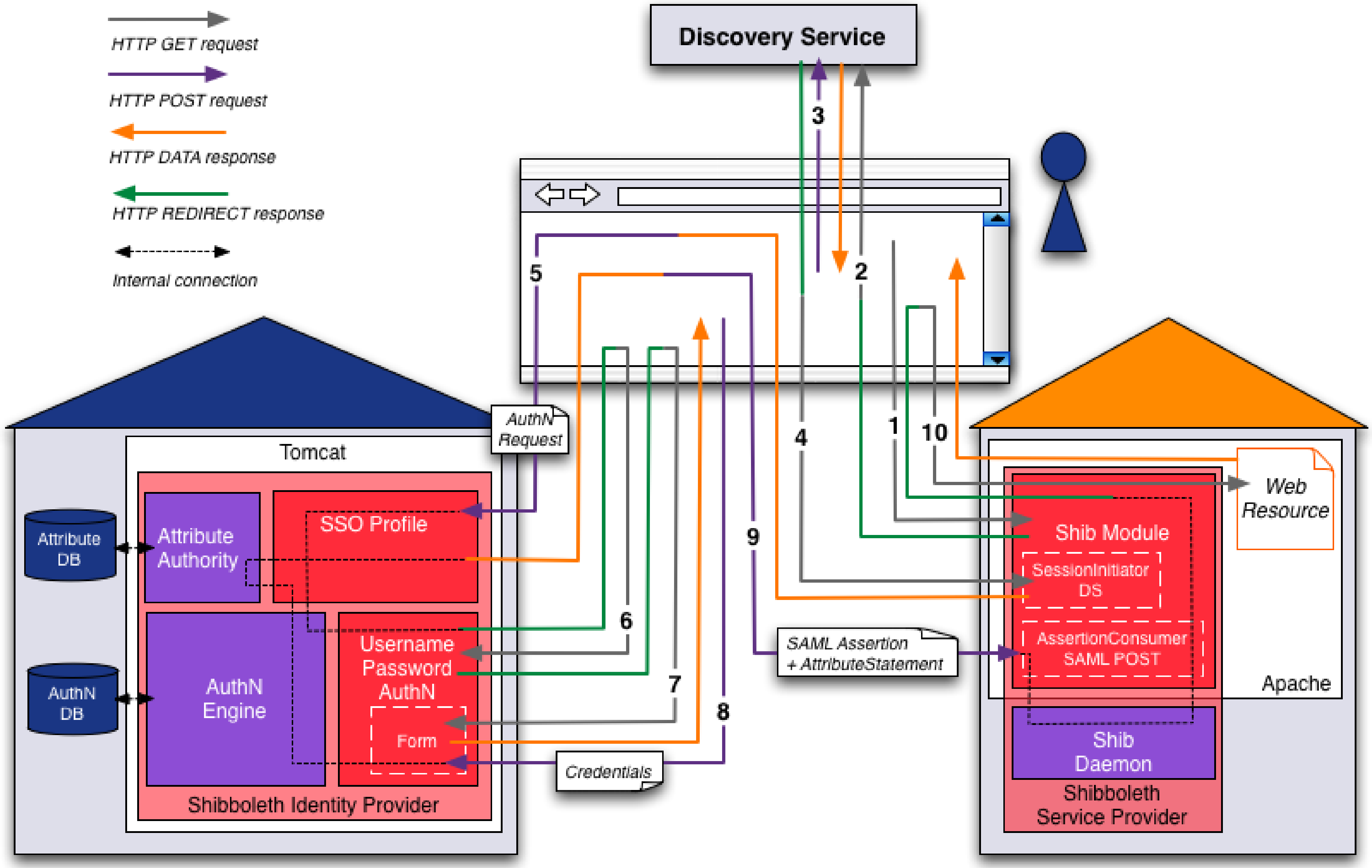
„Shibboleth is a ***standards based****,* ***open source*** *software package for* ***web single sign-on across or within organizational boundaries****.“*

- Began as an Internet2 activity in 2000
- SAML (2.0)
- Maintained by the Shibboleth Consortium
- Widespread useage in research & education
- Origin of the word: see the Bible, Judges 12,5.6

Shibboleth Project

- Identity Provider
- Service Provider
- OpenSAML Libraries (C++ & Java)
- Metadata Aggregator
- Centralized Discovery Service (discontinued, use SWITCH WAYF as an alternative)
- Embedded Discovery Service

WebSSO in Detail



(c) SWITCHaai

Shibboleth IdP

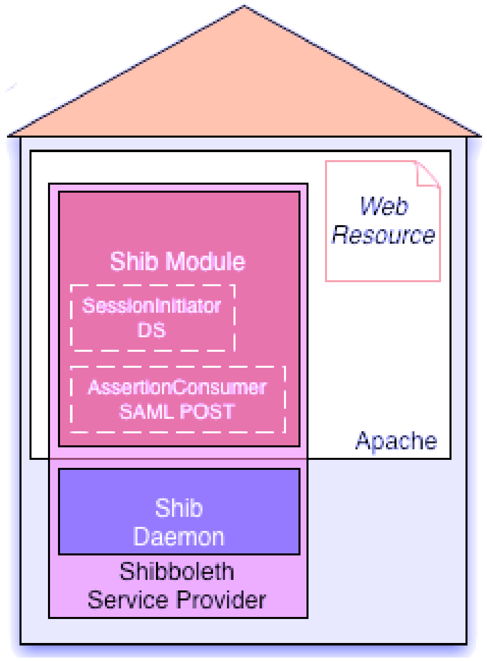
- Implements the Identity Provider (IdP) Component
- AuthN of users from an organisation happens centrally at its IdP
- Assertions (SAML) to convey identity informations (AuthN result + attributes) to Relying Parties (RP)
- Shibboleth IdP is a Java application
- Protocol Support includes SAML 1.1, most SAML 2.0 profiles, CAS
- Based on Spring for configuration & wiring of components
- Current release: V3.3.x

Shibboleth IdP

- Highly configurable
  - Authentication sources
  - Attribute Resolution
  - Attribute Filtering
  - Metadata Management
  - Per - Relying Party (i.e. per-SP) Configuration
  - Policies
  - Backend / Storage

Shibboleth SP

- Implements the SP component
- There are two parts
  - C++ daemon (shibd)
    - Keeps state, evaluates protocol mesages
  - Webserver module (mod_shib)
    - Protects Locations/Directories, defines Access Rules
- Current release: V3.0.x
- Binaries for Windows, OSX, RPM-based Linux
- SWITCH Repository for Ubuntu/Debian



(c) Switch

Shibboleth SP Overview

- General configuration in /etc/shibboleth/shibboleth2.xml

- Applications as „containers“ to group common configuration

- Configuration of <Session> properties (e.g. duration)

- <SessionInitiator> define how the SP should start a session (shorthands for e.g. <SSO> and <Logout> exist)

- <Handler> provide various endpoints

```
<ApplicationDefaults entityID="https://sp.org/shibboleth
       REMOTE_USER="eppn persistent-id targeted-id">

  <Sessions lifetime="28800" timeout="3600"
            relayState="ss:mem"
            checkAddress="false" handlerSSL="true"
            cookieProps="https">

    <SSO discoveryProtocol="SAMLDS"
        discoveryURL="https://wayf.org/ds">
      SAML2 SAML1
    </SSO>

    <Handler type="MetadataGenerator"
             Location="/Metadata" signing="false"/>

    <Handler type="Status" Location="/Status"
             acl="127.0.0.1 ::1"/>

    <Handler type="Session" Location="/Session"
             showAttributeValues="false"/>
```

Shibboleth SP Overview

- General configuration in /etc/shibboleth/shibboleth2.xml

- Load metadata (static or dynamic)
- Configure how the SP should provide attributes to applications
- Keys and Certificates
- Possible ApplicationOverrides to run multiple logical SPs with one installation

```
<MetadataProvider type="XML" validate="true"
    uri="http://example.org/fed-metadata.xml"
    backingFilePath="fed-metadata.xml"
    reloadInterval="7200">

    <MetadataFilter type="Signature"
        certificate="md_federation.pem"/>
</MetadataProvider>

<AttributeExtractor type="XML" validate="true"
    reloadChanges="false"
    path="attribute-map.xml"/>

<AttributeFilter type="XML" validate="true"
    path="attribute-policy.xml"/>

<CredentialResolver type="File"
    key="sp-key.pem"
    certificate="sp-cert.pem"/>

<!--<ApplicationOverride /> -->
</ApplicationDefaults>
```

Processing Attributes

- /etc/shibboleth/attribute-map.xml to extract attributes and provide them internally
- Simple mapping from incoming name to internal id

```
<Attribute name="urn:oid:0.9.2342.19200300.100.1.3" id="mail"/>
```

- You can then use them as apache environment variables in protected applications, e.g. `$_SERVER[‘mail‘]`
- /etc/shibboleth/attribute-policy.xml to define processing policies
- Default configuration includes e.g. scope checking of eduPersonPrincipalName, eduPersonScopedAffiliation, etc.

Filter Attributes

- /etc/shibboleth/attribute-policy.xml
- https://wiki.shibboleth.net/confluence/display/SP3/AttributeFilter

```
<afp:AttributeRule attributeID="sn">

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

Process Attributes

- Attributes can be changed various times on their way to the application
  - IdP resolves attributes
  - IdP filters attributes, usually on a per-SP basis
  - SP accepts or rejects attributes…
  - ...filters them accoring to policies…
  - ...and maps them to internal variables
  - Webserver provides variables as environment variables
- Be aware that names might change and attributes might get lost

Protecting Applications

- Three common options
  - Directly in the web server via Apache Access Rules

```
<Location /application>
  AuthType shibboleth
  ShibRequestSetting requireSession 1
  require shib-session
</Location>
```

  - In shibboleth2.xml configuration via SPXMLAccessControl
  - Use application („lazy session“ or „passive protection“)

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

DAASI International

Protecting Applications

- Usually Apache Access Rules is the easiest approach
- Require shib-attr allows to decide based on attributes

```
<Location /application>
  AuthType shibboleth
  ShibRequestSetting requireSession 1

   # use either:
  require shib-session

   # or something like:
  require shib-attr affiliation student


</Location>
```

Troubleshooting – SP Handlers

- `https://sp.example.org/Shibboleth.sso/Session`

  switch on showing attribute values:

```
  <Handler type="Session" Location="/Session"
           showAttributeValues="true"/>
```

- `.../Metadata`
- `.../Status`   - permit own IP address

```
  <Handler type="Status" Location="/Status"
acl="127.0.0.1 ::1 11.22.33.44"/>
```

Further SP Handlers

- `.../DiscoFeed` (JSON feed that contains all IdPs known to the SP, to be used in IdP discovery)
- `.../Login?param1=val1&param2=val2&...`
  - z.b. `target, entityID, forcedAuthn`
  - https://wiki.shibboleth.net/confluence/display/SP3/SessionInitiator
- `.../Logout?return=http://google.de`

Troubleshooting

- Firefox Plugin SAML Tracer
- Log Files
  - /var/log/shibboleth/shibd.log
  - /var/log/shibboleth/transaction.log (shows attributes)
  - /var/log/shibboleth/httpd/native.log (Apache's mod_shib Log)
  - /var/log/httpd/ssl_access_log und ssl_error_log
- Place a phpinfo() under /secure/index.php to see all variables, or /Shibboleth.sso/Session

Shibboleth SP Single Logout

- Complete implementation of the SAML standard
- Default: https://sp.example.org/Shibboleth.sso/Logout?return=https://sp.example.org/myApp/unauthenticated.php
- Konfiguration Options:
  - Trigger no Logout or only Local Logout
  - Trigger Asynchronous Logout (default): Do local logout and pass control to the IdP
  - Synchronous Logout: after IdP Logout, return control to SP
  - Back Channel Logout: use SOAP SLO endpoint of IdP (also Synchronous)
- Notify application using front channel or back channel

Shibboleth DS

- Two pieces still missing
- SP uses Discovery Service (DS) to allow users to choose their IdP

```
<SSO discoveryProtocol="SAMLDS"
     discoveryURL=https://example.org/ds">
  SAML2 SAML1
</SSO>
```

- Shibboleth *did* provide Centralized Discovery Service (CDS)
  - Central as in „not on every SP“
  - The federation operator might run this
  - Shibboleth CDS is no longer maintained
- Shibboleth does provide Embedded Discovery Service (EDS)
  - Embedded as in „runs on every SP host“
  - Can be combined with Metadata management of Shibboleth SP

DAASI International

Shibbolizing Applications

Apache Access Control

- Apache Access control in httpd.conf:

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

- Supports Regexps
- .htaccess files: only works for "real" directories
- Apache >=v2.4: Combine using boolean rules
- Apache < 2.4: use ShibRequireAll if needed
- https://wiki.shibboleth.net/confluence/display/SP3/htaccess

DAASI International

XML Access Control

- /etc/shibboleth/shibboleth2.xml (only option with IIS):

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

- Requires shibd restart (watch XML errors!)
- Only way to evaluate query parameters
- https://wiki.shibboleth.net/confluence/display/SP3/XMLAccessControl

DAASI International

Application Access Control

- Need at least lazy Session on the URL
- Can make use of arbitrary attributes
- E.g. PHP:

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

- E.g. in a Perl CGI Script:

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

- E.g. Java Servlet in Tomcat with Apache Proxy:

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

Lazy Session

- Application must implement access control
- Need passive protection:

```
<Location /lazy>
  AuthType shibboleth
  Require shibboleth
</Location>
```

- Application must trigger login:

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

Lazy Session: Option 2

- No need to call /Shibboleth.sso/Login directly
- Put the login URL under active protection and redirect there, when user wants to log in

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

- Use this part of the application to evaluate attributes and store them to the session.

ApplicationOverride

- Logical Service Provider that usually corresponds to some separate Apache Config:

```
<Location />
    ShibRequestSetting applicationId myappname
</Location>
```

- ...or shibboleth.xml request map for IIS:

```
<RequestMap applicationId="default">
...
    <Host name="myapphost.example.org"
          applicationId="myappname"/>
...
</RequestMap>
```

ApplicationOverride

- Define the logical application in shibboleth2.xml:

```
<ApplicationDefaults ...>
...
    <ApplicationOverride id="myappname">
        <Sessions lifetime="900" timeout="900"
                  checkAddress="false"
                  handlerURL="/MyAppShibboleth.sso" />
    </ApplicationOverride>
</ApplicationDefaults>
```

- Usages for overrides:
  - Different Session lengths
  - Different attribute mappings / policy
  - Different certificates / default IdPs
- https://wiki.shibboleth.net/confluence/display/SP3/ApplicationOverride

isPassive at the SP

- Do I have a session at the IdP? The SP cannot know without trying!
- Pass the Discovery Service (user must have her Cookie for the chosen IdP there)
- SAML request passes the IdP and returns to the SP silently:
  - Existing IdP session: with a SAML Assertion
  - No IdP session: with a SAML "error" response *noPassive.* **User does not see a login form!**
- SP knows how to handle *noPassive* and would show a default page
- isPassive at Shibboleth IdP depends on Authentication Mechanism, e.g. RemoteUser does not support it

forceAuthn

- Force Re-Authentication despite of SSO
- Can be used for critical (parts of) applications
- Either set in the <SSO> handler for forcing generally,
- Or force case-wise: evaluate Authn Timestamp and call...

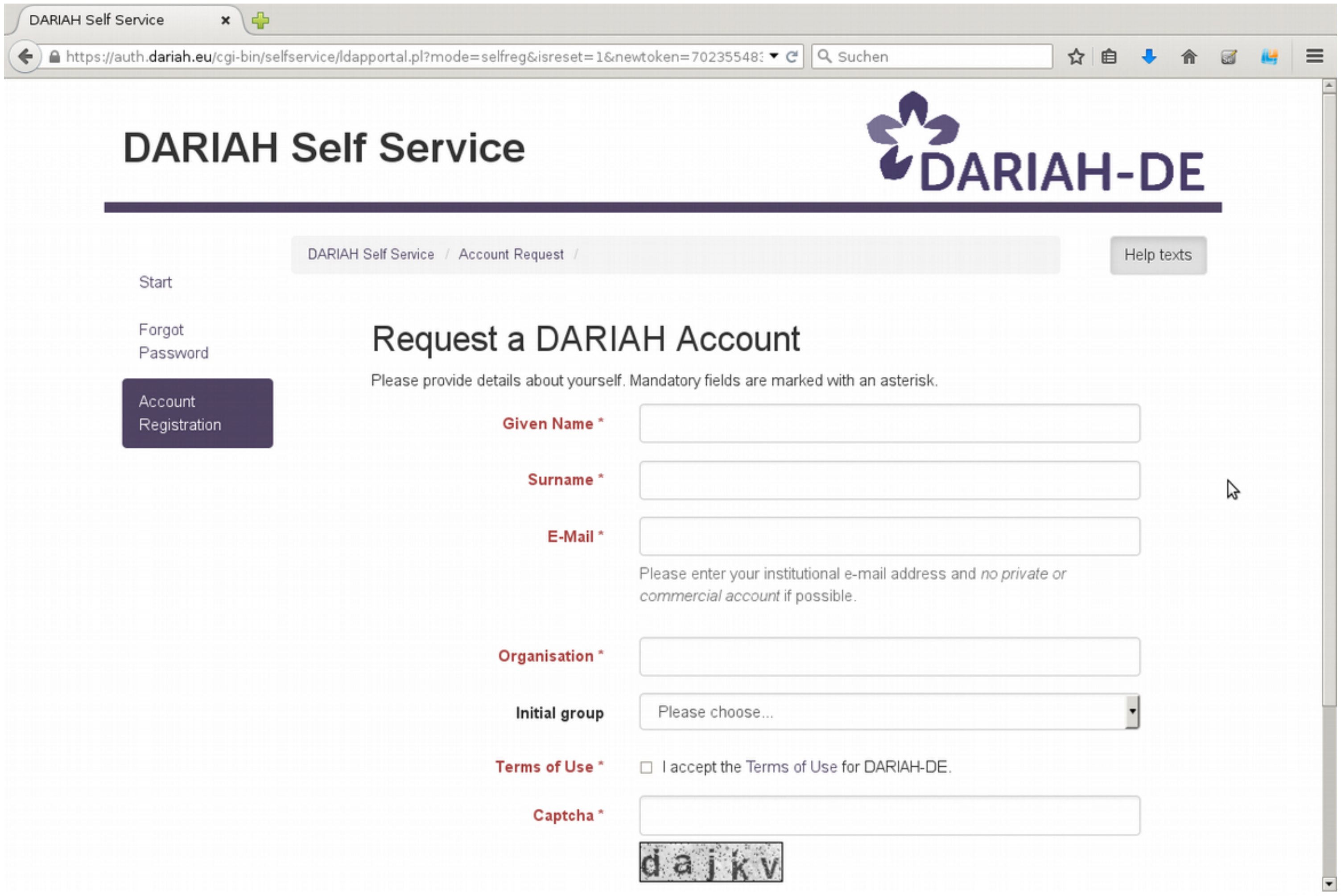
  /Shibboleth.sso/Login?forceAuthn=true

DARIAH AAI

DARIAH AAI - challenges

- Goal 1: users of DARIAH services (SPs) should authenticate via their home organization (campus IdP).
- Goal 2: certain DARIAH services only allow particular user groups. This should be configurable centrally by the respective admins, for all DARIAH services.
- Goal 3: DARIAH needs some user information
  - 3.a) she agrees to DARIAH Terms
  - 3.b) she is a researcher (e.g. her organization or e-mail)
- Goal 4: cope with a situation where users either
  - 4.a) have no campus IdP
  - 4.b) their campus IdP would not release Personally Identifiable Information (PII) to hitherto unknown SPs

DARIAH AAI - Self Service



DARIAH Self Service

https://auth.dariah.eu/cgi-bin/selfservice/ldapportal.pl?mode=selfreg&isreset=1&newtoken=70235548:

DARIAH Self Service

DARIAH-DE

DARIAH Self Service / Account Request /

Help texts

Start

Forgot Password

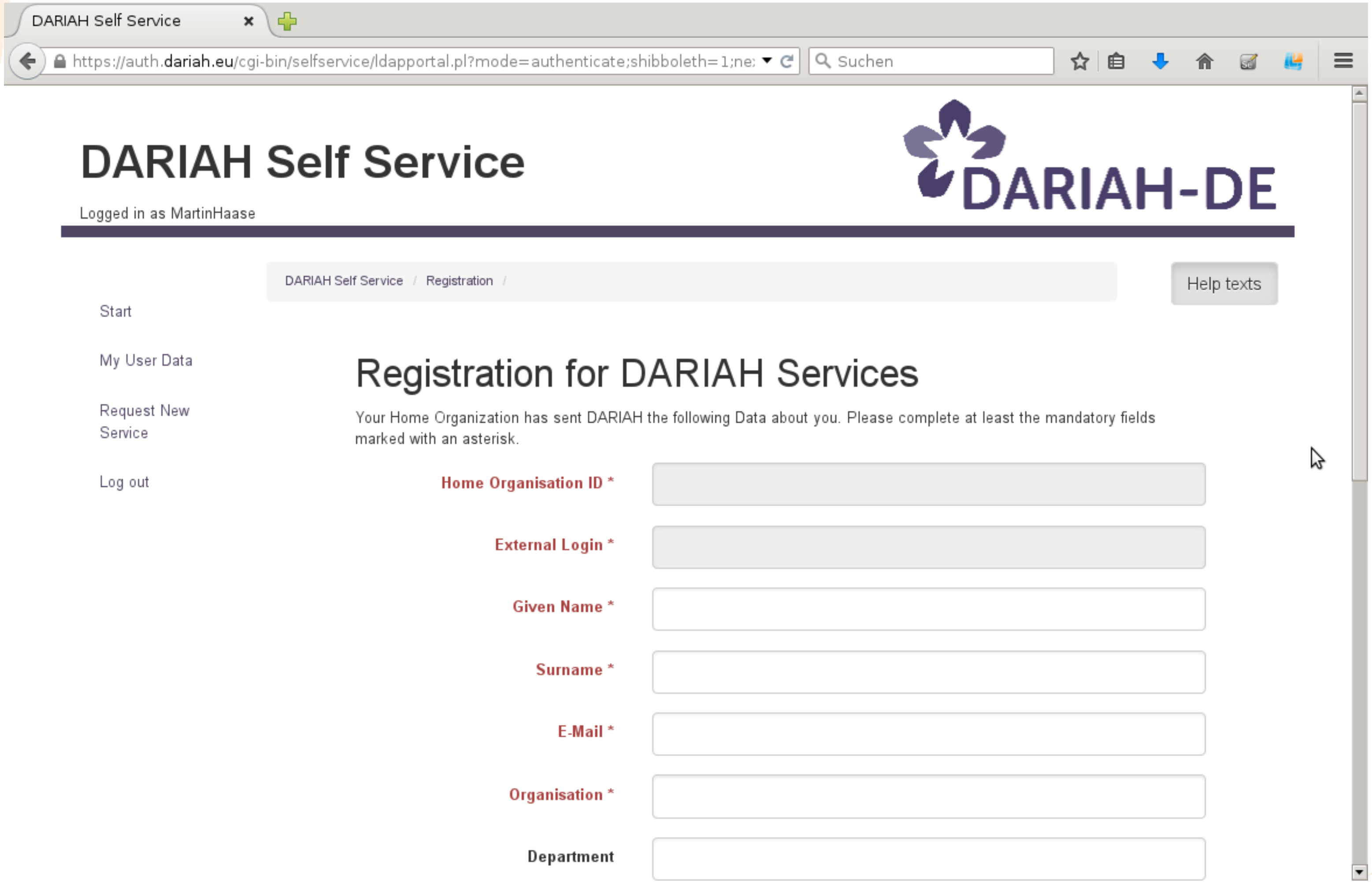
Account Registration

Request a DARIAH Account

Please provide details about yourself. Mandatory fields are marked with an asterisk.

Given Name *	
Surname *	
E-Mail *	Please enter your institutional e-mail address and *no private or commercial account* if possible.
Organisation *	
Initial group	Please choose...
Terms of Use *	☐ I accept the Terms of Use for DARIAH-DE.
Captcha *	

DARIAH AAI - Register Federation Users


DARIAH Self Service
https://auth.dariah.eu/cgi-bin/selfservice/ldapportal.pl?mode=authenticate;shibboleth=1;ne:
Suchen
DARIAH-DE
Help texts

DARIAH Self Service

Logged in as MartinHaase

DARIAH Self Service / Registration /

Start

My User Data

Request New Service

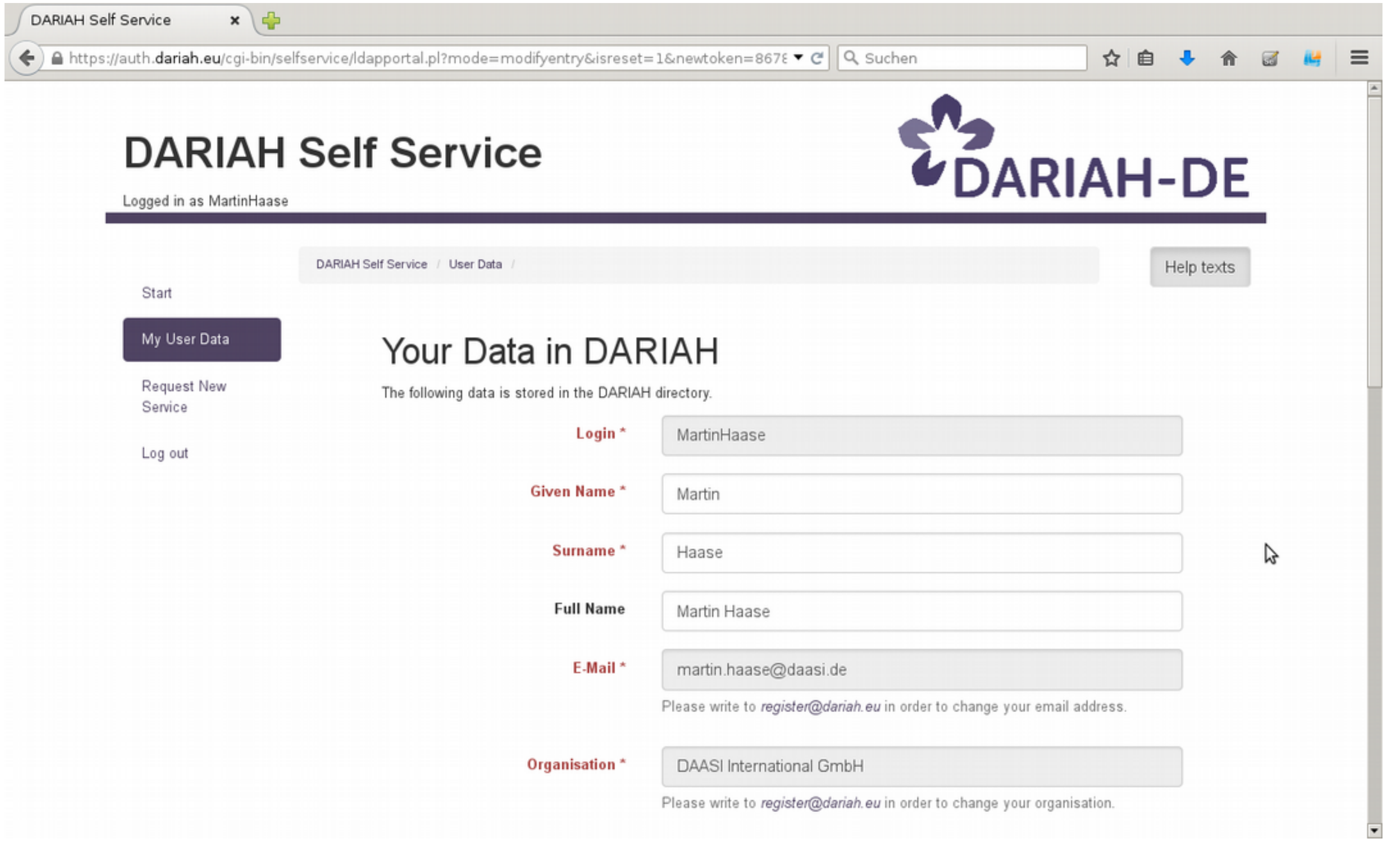
Log out

Registration for DARIAH Services

Your Home Organization has sent DARIAH the following Data about you. Please complete at least the mandatory fields marked with an asterisk.

Home Organisation ID *	
External Login *	
Given Name *	
Surname *	
E-Mail *	
Organisation *	
Department	

DARIAH AAI - Self Service


DARIAH Self Service
https://auth.dariah.eu/cgi-bin/selfservice/ldapportal.pl?mode=modifyentry&isreset=1&newtoken=8678
Suchen
DARIAH-DE

DARIAH Self Service

Logged in as MartinHaase

DARIAH Self Service / User Data /

Help texts

Start

My User Data

Request New Service

Log out

Your Data in DARIAH

The following data is stored in the DARIAH directory.

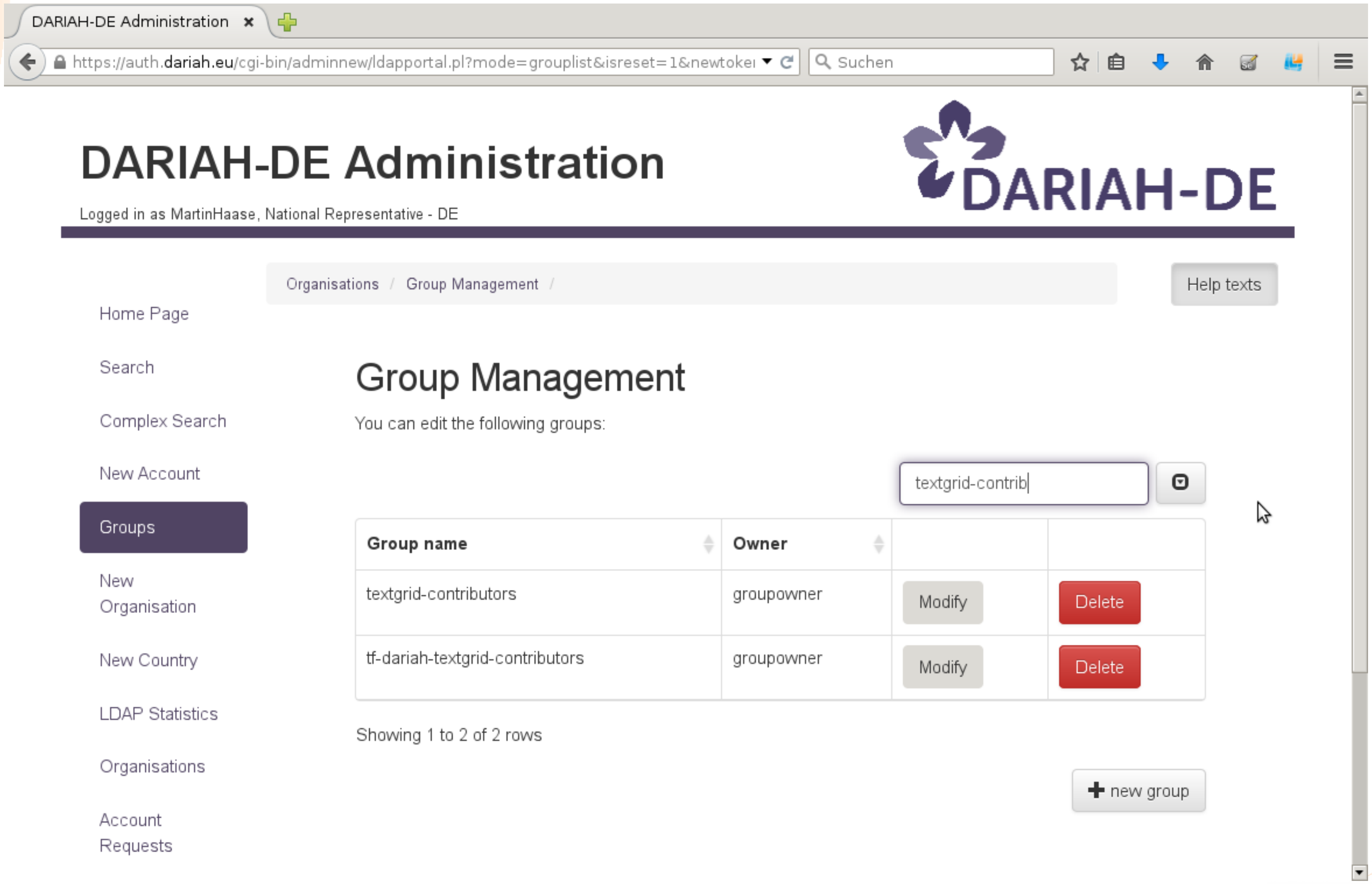
Login *	MartinHaase
Given Name *	Martin
Surname *	Haase
Full Name	Martin Haase
E-Mail *	martin.haase@daasi.de

Please write to *register@dariah.eu* in order to change your email address.

Organisation *	DAASI International GmbH

Please write to *register@dariah.eu* in order to change your organisation.

DARIAH AAI - Administration: AuthZ Groups


DARIAH-DE Administration

https://auth.dariah.eu/cgi-bin/adminnew/ldapportal.pl?mode=grouplist&isreset=1&newtoke

DARIAH-DE Administration

Logged in as MartinHaase, National Representative - DE

Organisations / Group Management /

Help texts

Home Page

Search

Complex Search

New Account

Groups

New Organisation

New Country

LDAP Statistics

Organisations

Account Requests

Group Management

You can edit the following groups:

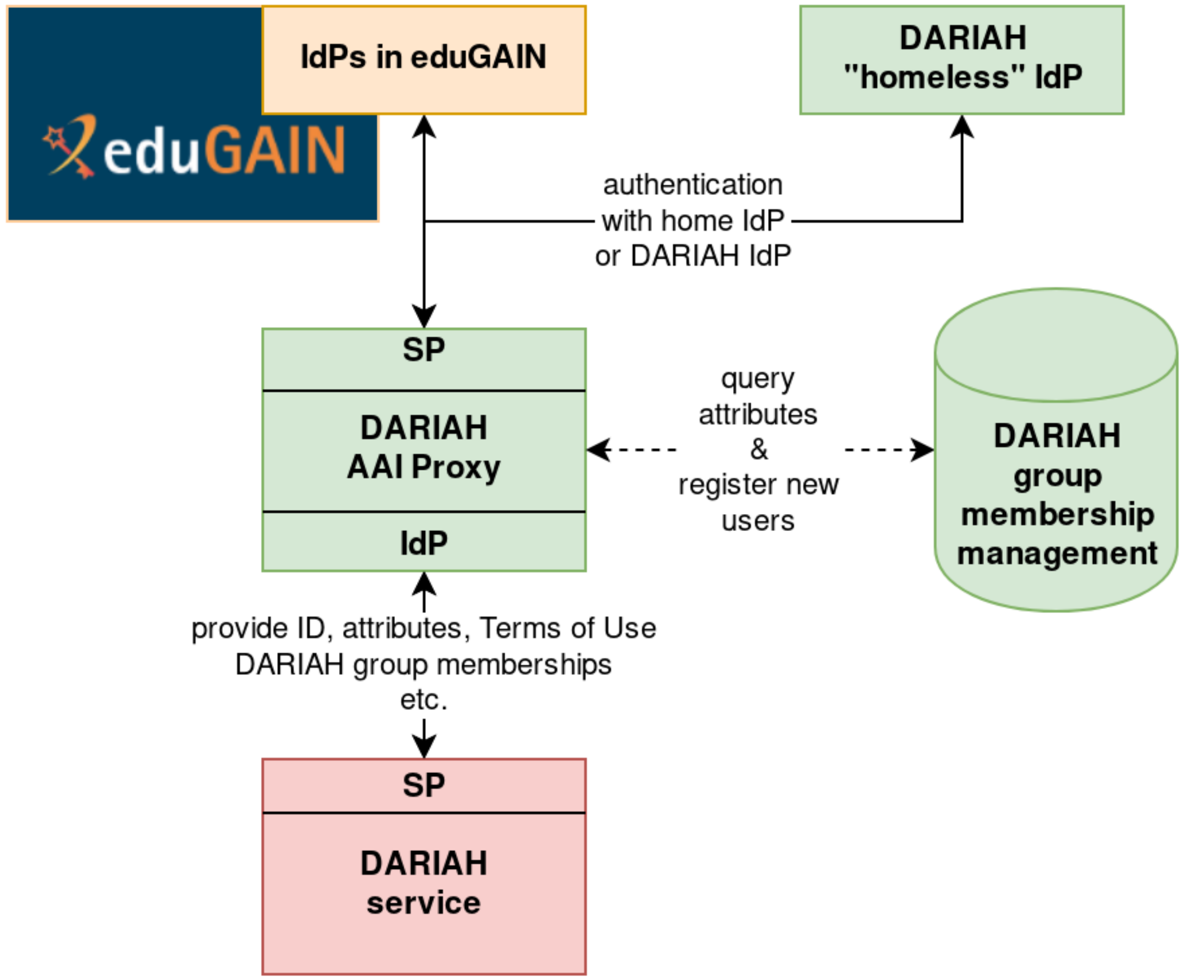
textgrid-contrib

Group name	Owner		
textgrid-contributors	groupowner	Modify	Delete
tf-dariah-textgrid-contributors	groupowner	Modify	Delete

Showing 1 to 2 of 2 rows

+ new group

DARIAH AAI - Overview


eduGAIN
IdPs in eduGAIN
DARIAH "homeless" IdP
authentication with home IdP or DARIAH IdP
SP
DARIAH AAI Proxy
IdP
query attributes & register new users
DARIAH group membership management
provide ID, attributes, Terms of Use DARIAH group memberships etc.
SP
DARIAH service

DAASI International

Other SAML SP Implementations

Shibboleth SP vs. Libraries

- Shibboleth SP is (albeit written in C++) language-independent; Communication with application via
  - Handler URLs
  - Apache Environment Variables (optionally HTTP Headers)
- Can be used with Apache and IIS
- There exist many language-specific SAML libraries for common programming languages

Recommended Open Source Libraries

- simpleSAMLphp for PHP (can be used also as an IdP and as an IdP-SP-Proxy)
- pySAML2 for Python (the "2" is important)
- Spring Security SAML for Java+Spring (based on OpenSAML-Java, like the Shibboleth IdP)
- *Actually, since the AAI proxy is there, any well-maintained SAML implementation that can deal with encrypted SAML assertions will do.*

SSO beyond SAML: OAuth2 & OIDC

OAuth2 Basics

- IETF Framework for Authorization (AuthZ)
- Use Case: A Client wants to programmatically access a particular user's resources living on a particular Resource Server (RS)
- Examples:
  - Access a user's profile (Social Media)
  - Securing APIs (usually REST-based ones)
- OAuth2 Core specifies
  - How AuthZ is gained (→Grants and Endpoints)
  - How AuthZ is represented (→ Access Tokens)
  - How expired Access Token are renewed (→ Refresh Tokens)
- Extensions exist, e.g. Token Introspection, ...

OAuth2 Roles

- Resource Owner (RO): a user that grants AuthZ
- Resource Server / Resource Provider (RS): a server that hosts the resource to be accessed
- Client: an application that wants to access these resources
- Authorization Server (AS): manages trust, issues tokens, offers token validation endpoints, asks for AuthZ, ((handles AuthN))

OAuth2 Terms

- Access Token
  - Form of an Access Token is not specified (e.g. random String vs. JSON token with internal structure)
  - Usually Bearer Tokens
- Refresh Token: transparently allow the client to request a new access token
- Scope: permissions bound to a token (read, write, email, photos, …)
  - Client can request a particular set of scopes
  - The AS decides which scopes to bind to the token

OAuth2 Bearer Tokens

- Anyone in possession of the Access Token is authorized
- AS issues token:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "access_token":"mF_9.B5f-4.1JqM",
  "token_type":"Bearer",
  "expires_in":3600,
  "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA"
}
```

- Client uses it:

```
GET /resource/r324/data HTTP/1.1
Host: server.example.com
Authorization: Bearer mF_9.B5f-4.1JqM
```

OAuth2 Flows

- Authorization Code Grant
  - For Web Applications (=Client is a Server-side application)
  - An authenticated client receives one-time *Authorization Code* via a Browser Redirect and exchanges it for an Access Token using a REST call to the AS
- Implicit Grant
  - For Mobile Applications (=JavaScript Client in the Browser etc.)
  - An unauthenticated client recieves the Access Token directly from the AS
- Resource Owner Password Credential Grant
  - Client uses credential of the RO directly to get a token
- Client Credential Grant
  - Only for authenticating the client

OAuth2 Endpoints

- Authorization Server Endpoints:
  - Authorization Endpoint
  - Token Endpoint
- Client Endpoints
  - Redirection Endpoint
- Extensions define further endpoints
  - e.G. Token Introspection

OAuth2 - just a Framework

- OAuth2 does not define AuthN (happens implicitly sometime and somehow)
- does not release information about the AuthN event
- does not release user attributes
- → not an SSO protocol
- → interoperability is limited and implementation-specific

OpenID Connect (OIDC)

- "simple identity layer on top of OAuth 2.0"
- Version 1.0, November 2014
- Support from companies like Google, Twitter, Facebook, ...
- Some say it outperforms SAML (...)
- Extends OAuth2 by
  - ID Tokens containing AuthN Information
  - UserInfo Endpoints to query for Claims (=Attributes)
- → comparable to Assertions in SAML

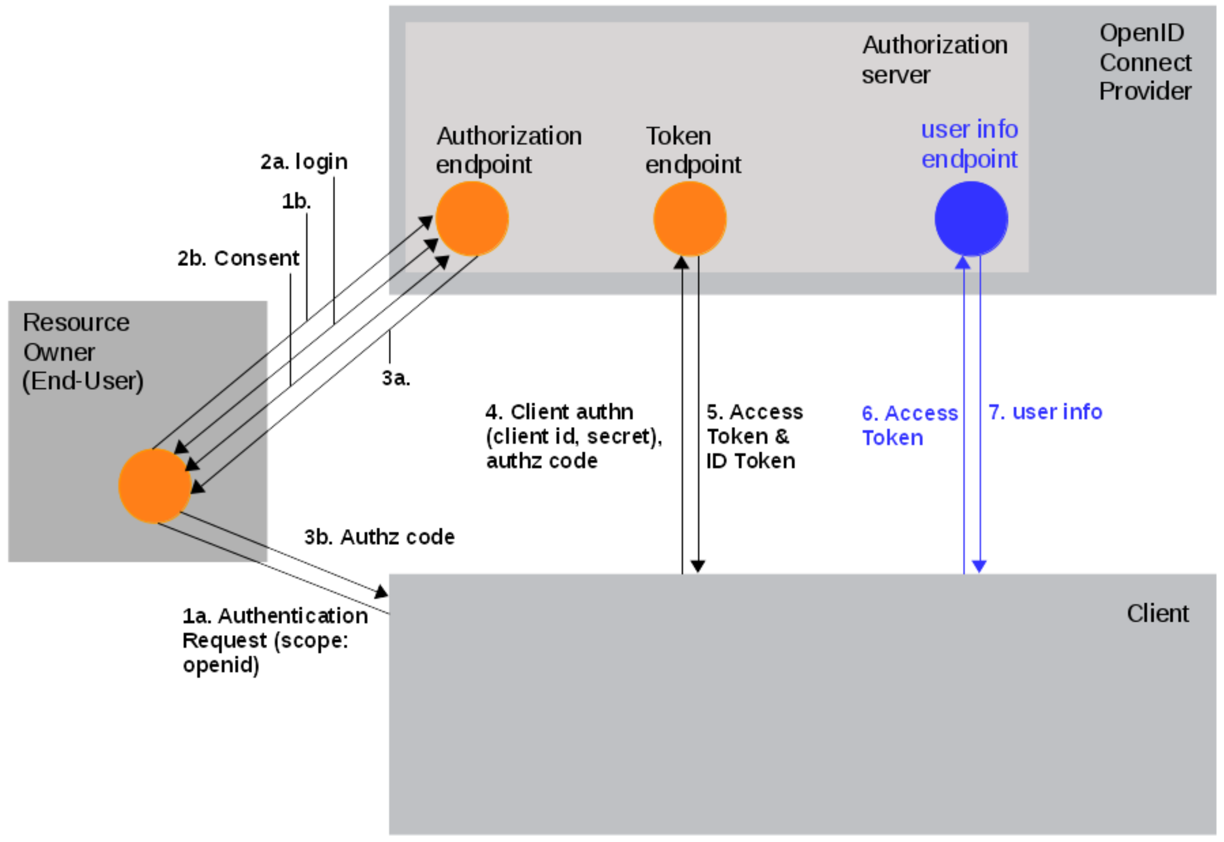
OIDC

- Re-use the OAuth2 Infrastructure
- Authentication Server (AS) is now an OIDC Provider (OP)
- Add "openid" as scope in client requests to the OP
- Allow for dynamic client registration (unlike static metadata in SAML)

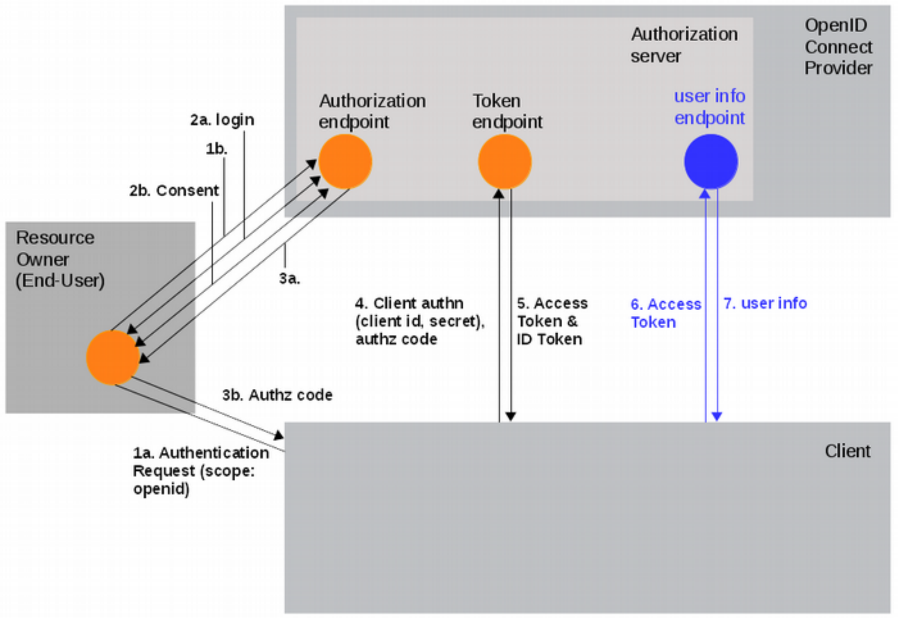
OIDC Authentication Flows

- Similar to OAuth2 Grant Types
- Authorization Code Flow
  - see Authorization Code Grant
- Implicit Flow
  - see Implicit Grant
- Hybrid Flow
  - Combination for applications consisting of front end and backend, which will each receive own tokens

Authorization Code Flow Example



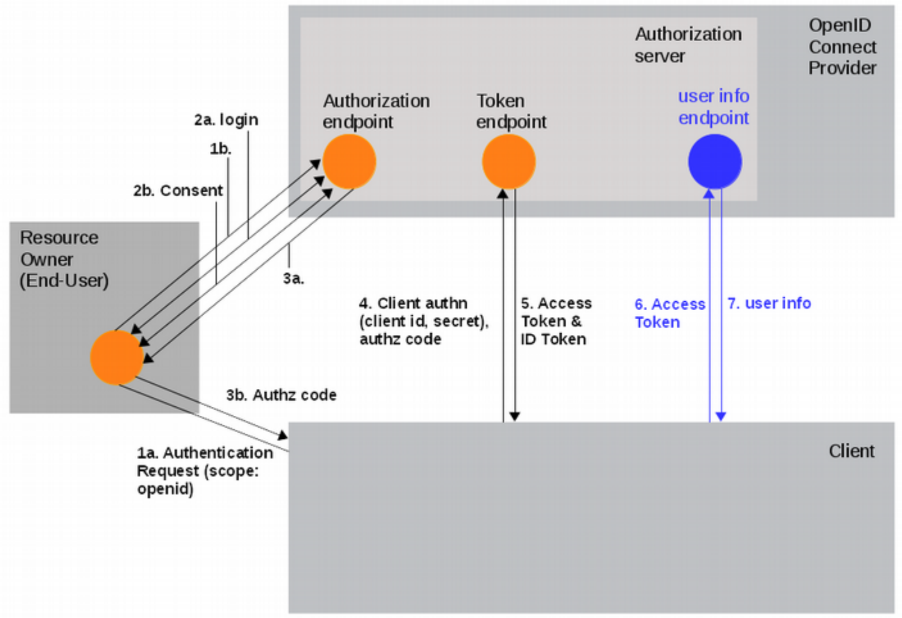
Authorization Code Flow



Step 1a

```
HTTP/1.1 302 Found
Location: https://server.example.com/authorize?
    response_type=code
    &scope=openid%20profile%20email
    &client_id=s6BhdRkqt3
    &state=af0ifjsldkj
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

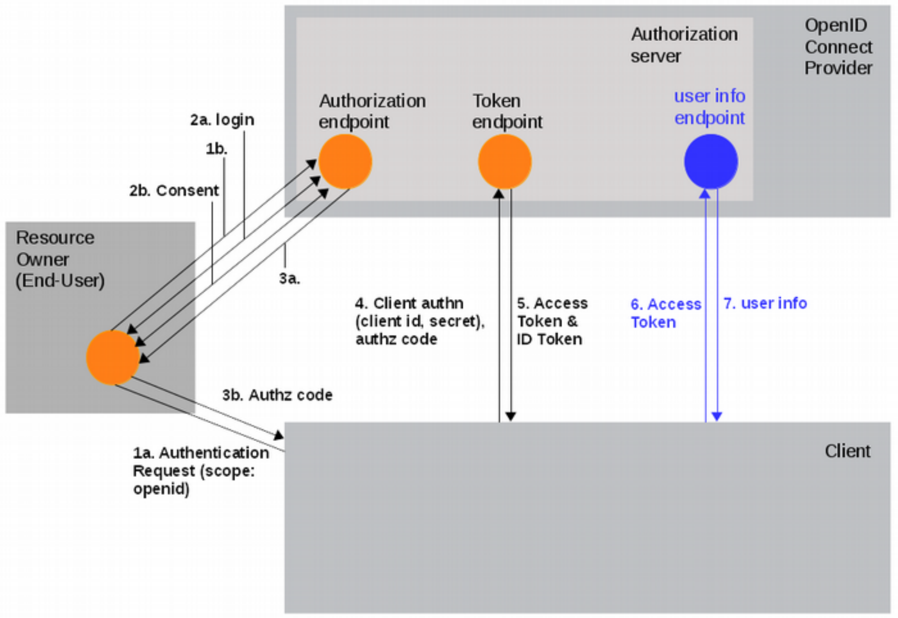
Authorization Code Flow



Step 1b

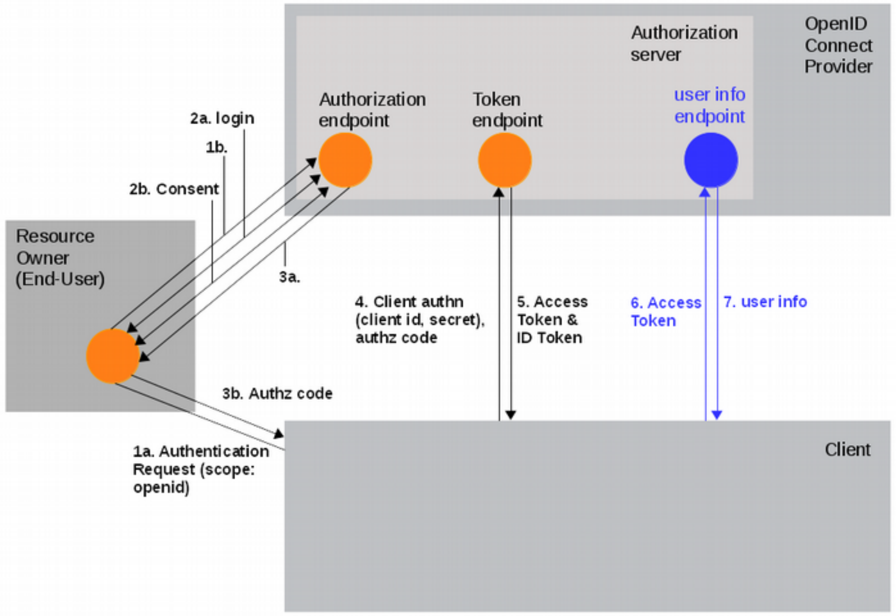
```
GET /authorize?
    response_type=code
    &scope=openid%20profile%20email
    &client_id=s6BhdRkqt3
    &state=af0ifjsldkj
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
HTTP/1.1
    Host: server.example.com
```

Authorization Code Flow



Step 2a & 2b: Login and Consent

Authorization Code Flow

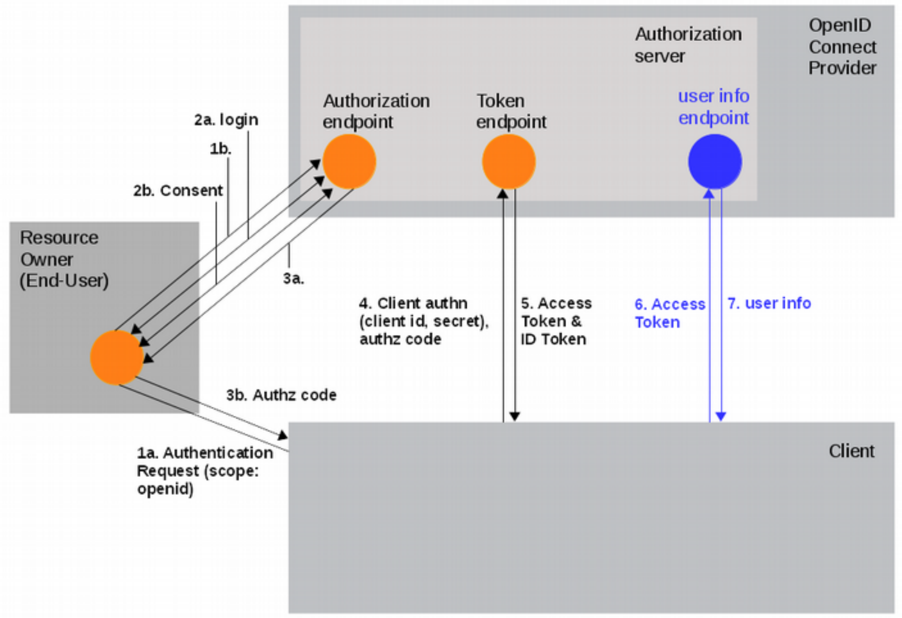


Step 3a & 3b

```
HTTP/1.1 302 Found
Location: https://client.example.org/cb?
    code=SplxlOBeZQQYbYS6WxSbIA
    &state=af0ifjsldkj

GET /cb?
    code=SplxlOBeZQQYbYS6WxSbIA
    &state=af0ifjsldkj HTTP/1.1
Host: client.example.com
```

Authorization Code Flow

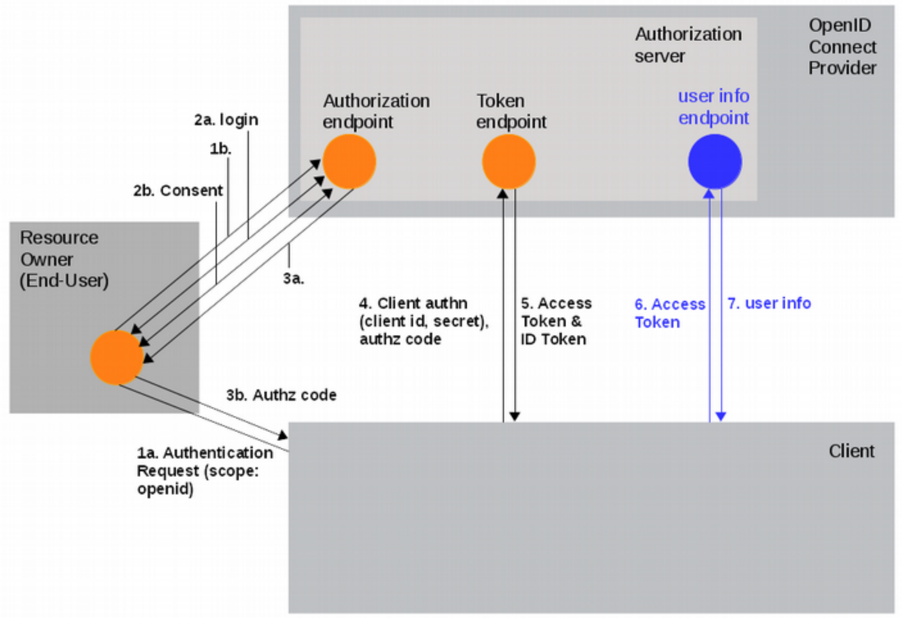


Step 4

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

Authorization Code Flow



Step 5

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
   "access_token": "SlAV32hkKG",
   "token_type": "Bearer",
   "refresh_token": "8xLOxBtZp8",
   "expires_in": 3600,
   "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI[...]"
}
```

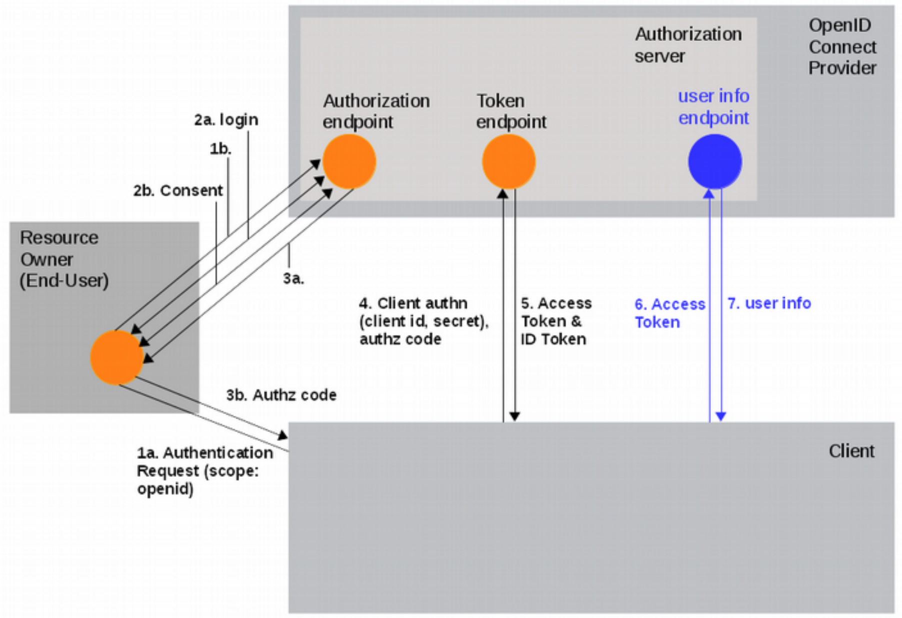
ID Token

- JSON Web Token
- Claims (=Attributes) about the user and about the AuthN event

```
{
   "iss": "https://server.example.com",
   "sub": "24400320",
   "aud": "s6BhdRkqt3",
   "nonce": "n-0S6_WzA2Mj",
   "exp": 1311281970,
   "iat": 1311280970,
   "auth_time": 1311280969,
   "acr": "urn:mace:incommon:iap:silver"
}
```

- Should be signed by the OP (JSON Web Signature), can be encrypted (JSON Web Encryption) sein

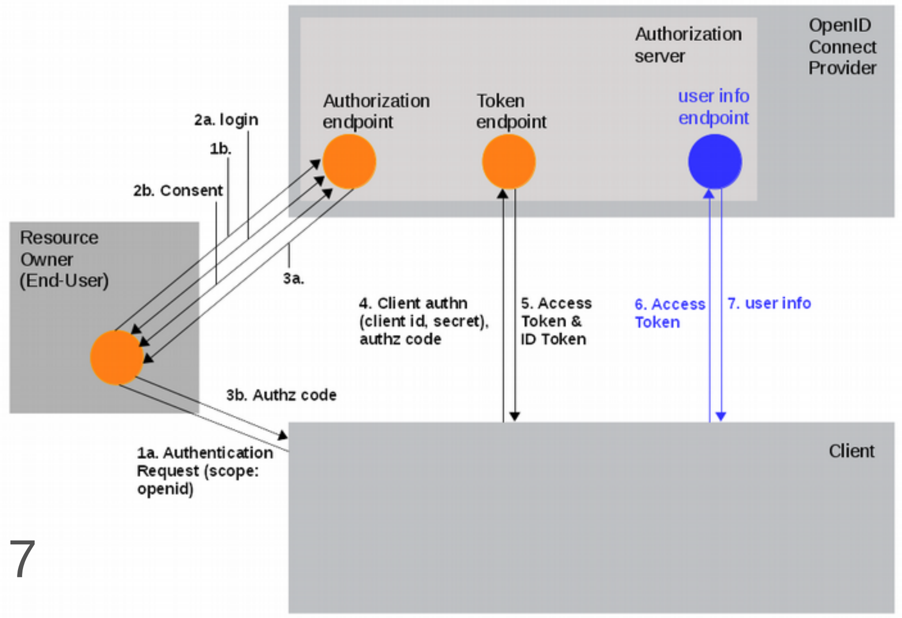
Authorization Code Flow



Step 6

```
GET /userinfo HTTP/1.1
Host: server.example.com
Authorization: Bearer SlAV32hkKG
```

Authorization Code Flow



Step 7

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "sub": "248289761001",
   "name": "Jane Doe",
   "given_name": "Jane",
   "family_name": "Doe",
   "preferred_username": "j.doe",
   "email": "janedoe@example.com",
   "picture": "http://example.com/janedoe/me.jpg"
}
```

OIDC Extensions

- Dynamic Client Registration
- Discovery
  - Find OP via WebFinger protocol
  - Retrieves OP's signing certificates, etc.
- Federation
  - like SAML federation
  - needs both discovery and dynamic client registration

Comparing OIDC und SAML

- JSON Web Token (JWT) vs. XML
- When communicating directly SP ↔ IdP: REST vs. SOAP
- OIDC: Browser-based JavaScript applications by design (Implicit Flow)
- SAML has the ECP Profile for securing (REST) APIs (only little uptake)
- Reasons for OIDC
  - More flexible for mobile and Javascript applications
  - Implementation on SP side easier
  - „lightweight“
- Reasons for SAML:
  - Well-established
  - Enables Federations
  - Multiple security measures inbuilt

Hands-On

SP Installation

- Installation instructions for all operating systems:
- https://www.switch.ch/aai/guides/sp/installation/

SP Configuration

- Set an entityID for your SP in /etc/shibboleth/shibboleth2.xml in the <ApplicationDefaults> element
- E.g. use an URL as the entityID, like so:
  - https://sp123.your.own.domain/shibboleth
  - https://localhost-for-john-doe-123/shibboleth
- Must be unique in the federation / among all SPs that your IdP(s) know

Connect the SP to the DARIAH AAI

https://wiki.de.dariah.eu/display/publicde/DARIAH+AAI+Documentation

Set up Trust with the AAI Proxy

- Download the DARIAH AAI Proxy's Metadata file like this (mind the capital "-O")

  ```
  wget https://aaiproxy.de.dariah.eu/idp -O
  /etc/shibboleth/dariah-proxy-idp.xml
  ```

- Locate the MetadataProvider section in /etc/shibboleth/shibboleth2.xml and add a row:

  ```
  <MetadataProvider type="XML" file="dariah-
  proxy-idp.xml"/>
  ```

- Send your own SP's metadata to register@dariah.eu (from `https://your.sp.edu/Shibboleth.sso/Metadata`)

Further SP Configuration

- In /etc/shibboleth/shibboleth2.xml, set the SP to direct login using

```
<SSO entityID="https://aaiproxy.de.dariah.eu/idp">
```

- Use the following ID preference order

```
REMOTE_USER="eppn unique-id"
```

- Enable eduPersonUniqueID in /etc/shibboleth/attribute-map.xml:

```
<Attribute name="urn:oid:1.3.6.1.4.1.5923.1.1.1.13"
           id="unique-id">

     <AttributeDecoder
           xsi:type="ScopedAttributeDecoder"/>

</Attribute>
```

Attributes in the DARIAH AAI

- Released by default: eppn, affiliation, unscoped-affiliation, entitlement
- Add and use unique-id for personalization (see previous slide)
- Configure in /etc/shibboleth/attribute-map.xml, by uncommenting: cn (CommonName), givenName, sn (surname), displayName, preferredLanguage, o (Organization), mail, schacCountryOfCitizenship
- DARIAH-specific Attributes, by adding:

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

<Attribute name="urn:oid:1.3.6.1.4.1.10126.1.52.5.2"
id="dariahRole"/>

<Attribute name="urn:oid:1.3.6.1.4.1.10126.1.52.4.15"
id="dariahTermsOfUse"/>
```

Attributes from Campus IdPs

- Adapt attribute-policy.xml to accept any scope
- Remove the Scope Checking stanza for affiliation and eppn

```
<afp:AttributeRule attributeID="affiliation">
            <afp:PermitValueRule xsi:type="AND">
                <RuleReference ref="eduPersonAffiliationValues"/>
<!-- accept any scope  <RuleReference ref="ScopingRules"/> -->
            </afp:PermitValueRule>
 </afp:AttributeRule> [...]
<!-- accept any scope for legacy users
        <afp:AttributeRule attributeID="eppn">
            <afp:PermitValueRuleReference ref="ScopingRules"/>
        </afp:AttributeRule> -->
```

Terms of Use for Service

- Generally DARIAH AAI ToU must be accepted by anyone using the DARIAH AAI
- The AAI proxy checks the ToU acceptance
- A DARIAH Service can have *additional* ToU
- Upload your ToU document to the DARIAH Repository, cf. DARIAH ToU:
- https://repository.de.dariah.eu → Documentation
- cf. https://dx.doi.org/10.20375/0000-000B-CB44-4

Terms of Use for Service (2)

- Then request an *authorization group* attached with your service's ToU:
- Log in to the SelfService, and choose
  - Manage Groups
  - "+ new group" (at the bottom of the page)
  - Choose a group name, e.g. ***myservice-users***
  - Put the link to your ToU in the "Remarks" box
- DARIAH staff will create the group for you

Thanks!

https://www.daasi.de