Umbrella Technical Documentation

- An Umbrella operators manual -

Table of Contents

1 Overview of Components

1.1 Introduction

Users of the photon and neutron facilities constitute a large community (with 30'000+ visiting scientists in Europe alone). At the moment the management of the experiments including identity management is performed locally on site via Web-based User Office (WUO) tools. However, users perform experiments increasingly at different facilities, about 30% on the average, in some research fields even up to 40%. They want to minimize the corresponding administration load and are interested in harmonized application surfaces. In addition, they need access to the data stored at the facilities and / or want to participate remotely in experiments. These trans-facility services need a federated identity management as provided by Umbrella. Furthermore, centralized data analysis will become more important which again needs a federated identity management.

Characteristic properties of Umbrella:

- Umbrella is not yet another identification system, but it is built on top (= umbrella) of the already existing Web-based User Office (WUO) systems of the participating large scale facilities with the additional functionality to enable a unique user identification.
- In order to guarantee uniqueness, Umbrella has only one (1) identity provider (IdP).
- User information is stored in a hybrid database system, where the central part contains the information (e.g. username + password) for user identification. All other authentication information and all authorization information remains at the local WUO systems.
- Umbrella is a bottom-up system. Authentication information is provided and updated via self-service by the user with optional confirmation loops with authorities. Supervision is provided by the user office staff. This avoids complex trust structures and procedures.
- Umbrella communication is based on SAML, which is state of the art, and is designed by industry experts like EMC, Hewlett Packard, IBM, Microsoft, Nokia, Oracle, SAP, Boeing et. al. [OASWS] It is also used by national federations in the (higher) education sector.
- The Umbrella user identity is persistent. It is not fixed to the home affiliation of the user which permits a permanent link of a user to a team or a dataset or document, also in case of an affiliation change.
- Umbrella is web-based and supports single sign on (SSO) functionality.
- Physically this one IdP is not be realized via only one central Umbrella server; there will be a geoDNS replication system in order to increase the uptime of the system and also to reflect the federal character of the system.

[UMBWS]

1.2 Setup of the System

As an introduction to the Umbrella system following illustration should give you a basic overview of all components involved and their physical setup:

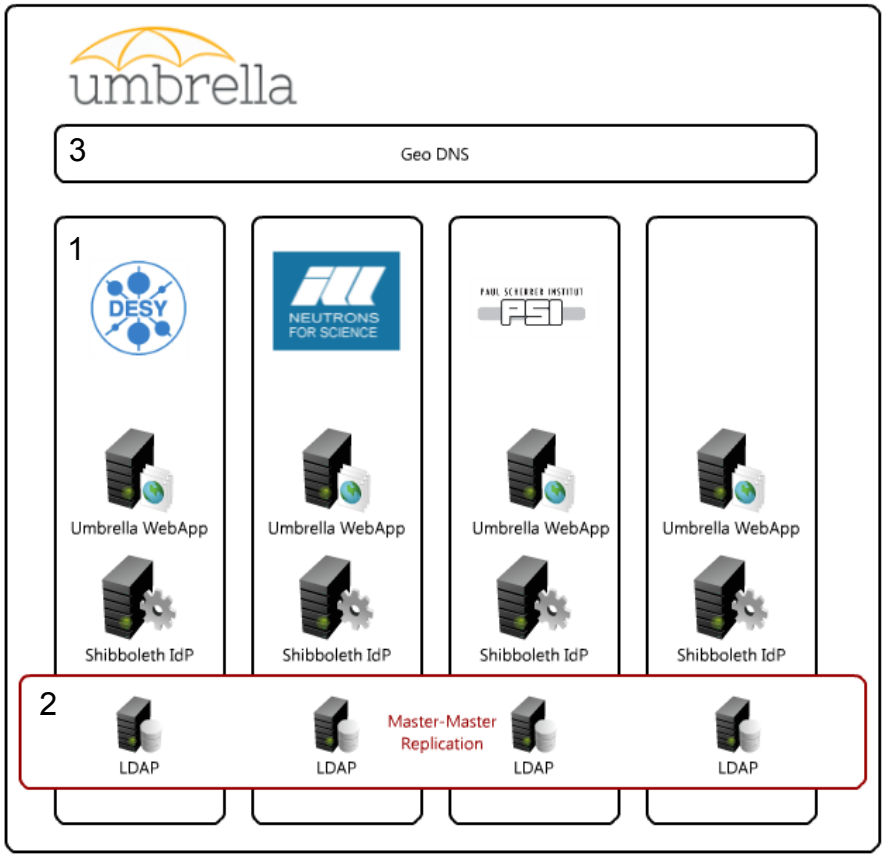


Illustration 1: Physical Umbrella setup

The illustration 1 shows:

- The components involved at one facility in the box 1 with the DESY logo: LDAP, Shibboleth IdP and Umbrella Webapp. The same setup is replicated for each facility running an Umbrella IdP instance.
- The master-master replication setup in-between the facilities in the red box with the number 2.
- The GeoDNS service enabling geographic distribution of the network load, sticky session and failover functionality. This service is shown with point 3 in the illustration.

2 Multi-master LDAP

Illustration 2: LDAP Multi-master replication

LDAP (Lightweight Directory Access Protocol) is an open vendor-neutral, industry standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol network [RFC4511].

In Umbrella it is used as a store to persist and replicate user definitions and attributes.

The LDAP server itself must be able to replicate in a master-master mode and as this functionality is not described in RFC4511 we must rely on a vendor-specific legacy solution to supply this functionality.

OpenDJ [ODJWS] was chosen as the implementing software for following reasons:

- Ease of use.
- Simplicity of installation and configuration.
- Asynchronous multi-master replication. Allows massive scalability over a Wide Area Network.
- Conflict resolution at entry and attribute level.

2.1 Local installation

The first step in installing a LDAP server is to download the software from here [ODJDS] and unpack it.

Please make sure to use the same version of the software in all installations to avoid version conflicts.

```
# cd /opt
/opt# wget http://download.forgerock.org/downloads/opendj/2.5.0-
Xpress1/OpenDJ-2.5.0-Xpress1.zip
/opt# unzip OpenDJ-2.5.0-Xpress1.zip
```

After unpacking the software it must be installed and configured with following command:

```
/opt# cd OpenDJ-2.5.0-Xpress1
/opt/OpenDJ-2.5.0-Xpress1# ./setup -i -n -a \
-b "dc=umbrellaid,dc=org" \
-h <YOUR_HOST_NAME> \
-p 389 \
--adminConnectorPort 4444 \
-D "<ADMIN_CREDENTIALS>" \
-w "<ADMIN_PASSWORD>"
```

Make sure to replace the values <YOUR_HOST_NAME>, <ADMIN_CREDENTIALS> and <YOUR_PASSWORD> before you issue this command.

This should leave you with an installed OpenDJ server in the `/opt` directory.

2.2 Configuration

2.2.1 Extending the Schema

To be able to store the Umbrella relevant information about a user the schema must be extended:

```
/opt/OpenDJ-2.5.0-Xpress1# cd config/schema
/opt/OpenDJ-2.5.0-Xpress1/config/schema# wget https://github.com/Umbrella-Commiters/UmbrellaIdP/raw/master/UmbrellaIdP/schema/99-user.ldif
```

Restart OpenDJ to reflect the changes

```
/opt/OpenDJ-2.5.0-Xpress1# bin/stop-ds
/opt/OpenDJ-2.5.0-Xpress1# bin/start-ds
```

2.2.2 Maximum Open Files

OpenDJ needs to be able to open many file descriptors, especially with many clients connected to it. Make sure OpenDJ can use at least 64K file descriptors. So if you run the OpenDJ server as user opendj you can set the hard and soft limit in the file `/etc/security/limits.conf`:

```
opendj soft nofile 65536
opendj hard nofile 131072
```

To check the overall availability of file descriptors you can issue the following command:

```
# cat /proc/sys/fs/file-max
201860
```

2.2.3 Java Settings

If you need high performance for production settings, following JVM options can be used. Please edit the entry `start-ds.java-args` in the file `config/java.properties` and apply the changes with the `dsjavaproperties` command:

```
/opt/OpenDJ-2.5.0-Xpress1# bin/dsjavaproperties
```

JVM Option	Description
`-server`	Use the C2 compiler and optimizer.
`-d64`	To use a heap larger than about 3.5 GB on a 64-bit system, use this option.

JVM Option	Description
`-Xmx,-Xmx`	Set both minimum and maximum heap size to the same value to avoid resizing. Leave space for the entire DB cache and more.
`-Xmn`	Set the new generation size between 1-4 GB for high throughput deployments, but leave enough overall JVM heap to avoid overlaps with the space used for DB cache.
`-XX:MaxTenuringThreshold=1`	Force OpenDJ to create only objects that have either a short lifetime, or a long lifetime.
`-XX:+UseConcMarkSweepGC`	The CMS garbage collector tends to give the best performance characteristics. You might also consider the G1 garbage collector.
`-XX:+PrintGCDetails -XX:+PrintGCTimeStamps`	Use these when diagnosing JVM tuning problems. You can turn them off when everything is running smoothly.
`-XX:+UseCompressedOops`	Java object pointers normally have the same size as native machine pointers. If you run a small, but 64-bit JVM, then compressed object pointers can save space. Set this option when you have a 64-bit JVM, -Xmx less than 32 GB, and Java SE 6u23 or later.

2.3 Operations

2.3.1 Start and Stop LDAP server

To start and stop the LDAP server following commands can be issued

```
/opt/OpenDJ-2.5.0-Xpress1# bin/stop-ds
/opt/OpenDJ-2.5.0-Xpress1# bin/start-ds
```

2.3.2 Init script

There is a script to help you creating an init-script located in the `bin` directory. The option `-f` defines the location and name of the script and the `-u` defines the user under which the server should run.

```
/opt/OpenDJ-2.5.0-Xpress1# bin/create-rc-script -f /etc/init.d/opendj -u
opendj
```

Add the script to the runlevels it should run under. Depending on your distribution there are commands to automate this:

```
Debian based:
# update-rc.d opendj defaults
```

```
RedHat based:
# chkconfig --level 35 opendj on
```

2.3.3 Enable LDAP replication on host

To enable the replication between hosts please gather following information in advance and make sure the server is installed and configured on both hosts:

Name	Value
YOUR_HOST	Your old existing first LDAP server. DNS names are preferred over IP addresses.
ADMIN_CREDENTIALS	Distinguished name of the administrator of the old LDAP server.
ADMIN_PASSWORD	The password of the administrator of the old LDAP server.
YOUR_NEW_HOST	The new LDAP server to be added to the replication topology.
NEW_ADMIN_CREDENTIALS	Distinguished name of the administrator of the new LDAP server.
NEW_ADMIN_PASSWORD	The password of the administrator of the new LDAP server.

Issue following command:

```
/opt/OpenDJ-2.5.0-Xpress1# bin/dsreplication enable \
--host1 <YOUR_HOST> \
--port1 4444 \
--bindDN1 "<ADMIN_CREDENTIALS>" \
--bindPassword1 <ADMIN_PASSWORD> \
--replicationPort1 8989 \
--host2 <YOUR_NEW_HOST> \
--port2 4444 \
--bindDN2 "<NEW_ADMIN_CREDENTIALS>" \
--bindPassword2 <NEW_ADMIN_PASSWORD> \
--replicationPort2 8989 \
--adminUID "<ADMIN_CREDENTIALS>" \
--adminPassword <ADMIN_PASSWORD> \
--baseDN "dc=umbrellaid,dc=org" \
-X -n
```

Please be aware that enabling the replication does not initialise the repositories in-between with data. This procedure is described under “Initialise host in LDAP replication”.

2.3.4 Initialise host in LDAP replication

To initialise a host in the replication topology the host must first be enabled to be part of the replication. This procedure is described in “Enable LDAP replication on host”.

Please gather following information prior to execute the command:

Name	Value
ADMIN_CREDENTIALS	Distinguished name of the administrator of the old LDAP server.
ADMIN_PASSWORD	The password of the administrator of the old LDAP server.
YOUR_HOST	Your old existing first LDAP server. DNS names are preferred over IP addresses.
YOUR_NEW_HOST	The new LDAP server to be initialised.

The initialisation is responsible to replicate all existing entries.

```
/opt/OpenDJ-2.5.0-Xpress1# bin/dsreplication initialize \
--baseDN "dc=umbrellaid,dc=org" \
--adminUID “<ADMIN_CREDENTIALS> \
--adminPassword <ADMIN_PASSWORD>  \
--hostSource <YOUR_HOST> \
--portSource 4444  \
--hostDestination <YOUR_NEW_HOST> \
--portDestination 4444 \
-X -n
```

After issuing this command the progress of the replication is shown and assures you that the replication is initialised.

To test if the replication is set-up correctly you can add a new entry on each side and verify if these show up on the other side. Then delete the entries and verify that they are deleted on the other side as well.

Now the replication topology is set-up. To add another new host to the replication topology go through the steps “Enable LDAP replication on host” and “Initialise host in LDAP replication”.

2.3.5 Remove node from LDAP replication

There might be several reasons why a node must be removed from the replication topology:

- A facility is no longer interested in running this services
- A facility is too unreliable
- A server gets hacked or is infected with malware

Following information is needed:

Name	Value
YOUR_HOST	The host to remove from the replication topology
ADMIN_CREDENTIALS	Distinguished name of the administrator of the LDAP server to be removed.
ADMIN_PASSWORD	The password of the administrator of the LDAP server to be

Name	Value
	removed.

To disable and remove a node from the replication topology following command can be used from any host which is part of the replication infrastructure:

```
/opt/OpenDJ-2.5.0-Xpress1# bin/dsreplication disable \
--hostname <YOUR_HOST> \
--port 4444 \
--adminUID <ADMIN_CREDENTIALS> \
--adminPassword <ADMIN_PASSWORD> \
--baseDN "dc=umbrellaid,dc=org" \
--disableReplicationServer \
-X -n
```

After this command the replication is completely removed from the specified host. All other hosts are still part of the replication topology.

2.3.6 Local backup

Backing-up the directory data is an important procedure which should happen on a daily basis. The procedure can either be done with the directory online or offline. The difference here is that if we use an online backup we need to authenticate and for and offline backup it is sufficient if the user can access the directory database files on the filesystem while the directory server is shutdown.

Following information is needed to do a full backup:

Name	Value
ADMIN_CREDENTIALS	Distinguished name of the administrator of the old LDAP server.
ADMIN_PASSWORD	The password of the administrator of the old LDAP server.
PATH_TO_BACKUP	Filesystem path on the directory server where the backup will be located.

Following command will initiate an immediate backup of everything and can be easily integrated into cron:

```
/opt/OpenDJ-2.5.0-Xpress1# bin/backup
 --port 4444 \
 --bindDN "<ADMIN_CREDENTIALS>" \
 --bindPassword <ADMIN_PASSWORD> \
 --backUpAll \
 --backupDirectory <PATH_TO_BACKUP> \
 --start 0
```

Normally we would like to schedule a full backup and the directory server offers an in-built function using the crontab format for scheduling jobs.

Following additional information is required:

Name	Value
<SUCCESS_EMAIL>	Email receiver if backup succeeds
<FAILURE_EMAIL>	Email receiver if backup fails

This command would schedule a full backup at 2.00 AM every night:

```
/opt/OpenDJ-2.5.0-Xpress1# bin/backup \
 --port 4444 \
 --bindDN "<YOUR_CREDENTIALS>" \
 --bindPassword <YOUR_PASSWORD> \
 --backUpAll \
 --backupDirectory <PATH_TO_BACKUP> \
 --recurringTask "00 02 * * *" \
 --completionNotify <SUCCESS_EMAIL> \
 --errorNotify <FAILURE_EMAIL>
```

At this point we should have a running backup and we should also regularly check if we are able to restore the backup.

To restore a backup we first need to stop the server and find its backup id. Following command will list all available backups of the userRoot. The ids are marked red:

```
/opt/OpenDJ-2.5.0-Xpress1# bin/stop-ds
/opt/OpenDJ-2.5.0-Xpress1# bin/restore \
--backupDirectory=<PATH_TO_BACKUP>/userRoot/ \
--listBackups

Backup ID:           20140630131937Z <-- <BACKUP_ID>
Backup Date:         30/Jun/2014:13:19:37 +0000
Is Incremental:      false
Is Compressed:       false
Is Encrypted:        false
Has Unsigned Hash:   false
Has Signed Hash:     false
Dependent Upon:      none

Backup ID:           20140630132558Z <-- <BACKUP_ID>
Backup Date:         30/Jun/2014:13:25:58 +0000
Is Incremental:      false
Is Compressed:       false
Is Encrypted:        false
Has Unsigned Hash:   false
Has Signed Hash:     false
Dependent Upon:      none
```

With this id we can now restore the specific backup and start the directory server again by issuing following commands:

```
/opt/OpenDJ-2.5.0-Xpress1# bin/restore \
--backupDirectory <PATH_TO_BACKUP>/userRoot/ \
--backupID <BACKUP_ID>
```

```
/opt/OpenDJ-2.5.0-Xpress1# bin/start-ds
```

2.3.7 Logfiles

Logfiles are an important source of information if we need to debug a specific behavior of the server. Following logfiles are available:

Path	Description
logs/access	The access log traces the operations the server processes including timestamps, connection information, and information about the operation itself. The access log can therefore grow quickly, as each client request results in at least one new log message.
logs/errors	The errors log traces server events, error conditions, and warnings, categorized and identified by severity.
logs/replication	The replication log traces all replication events, failures and warnings categorized and identified by severity.

2.3.8 Monitoring

Monitoring the current state of the LDAP server can be accomplished using different protocols:

- LDAP
- SNMP
- JMX

LDAP

OpenDJ exposes monitoring information over LDAP under the entry `cn=monitor`. The following example shows monitoring information about the `userRoot` backend:

```
/opt/OpenDJ-2.5.0-Xpress1# bin/ldapsearch \
--port 389 \
--baseDN cn=monitor "(cn=userRoot backend)"

dn: cn=userRoot backend,cn=Disk Space Monitor,cn=monitor
disk-state: normal
objectClass: top
objectClass: ds-monitor-entry
objectClass: extensibleObject
disk-free: 8682582016
disk-dir: /opt/OpenDJ-2.5.0-Xpress1/db/userRoot
cn: userRoot backend

dn: cn=userRoot Backend,cn=monitor
objectClass: top
objectClass: ds-monitor-entry
objectClass: ds-backend-monitor-entry
ds-backend-is-private: FALSE
ds-backend-writability-mode: enabled
```

```
cn: userRoot Backend
ds-backend-entry-count: 0
ds-base-dn-entry-count: 0 dc=umbrellaid,dc=org
ds-backend-id: userRoot
ds-backend-base-dn: dc=umbrellaid,dc=org
```

SNMP

OpenDJ lets you monitor the server over the Simple Network Management Protocol (SNMP), with support for the Management Information Base described in RFC2605 [RFC2605]

OpenDJ SNMP-based monitoring depends on OpenDMK, which you must download separately. Install the Full Binary Bundle by using the graphical installer, which requires that you accept the Binary License for Project OpenDMK. OpenDJ directory server that you download from ForgeRock is built with OpenDMK, but due to licensing OpenDMK is not part of OpenDJ. SNMP is therefore not enabled by default.

```
/opt/OpenDJ-2.5.0-Xpress1# cd /tmp
/tmp# wget https://opendmk.java.net/download/opendmk-1.0-b02-bin-dual-01-Oct-2007_19-17-46.jar
/tmp# java -jar opendmk-1.0-b02-bin-dual-01-Oct-2007_19-17-46.jar
```

If you install under `/opt`, then the runtime library needed for SNMP is `/opt/OpenDMK-bin/lib/jdmkrt.jar`.

Once you have installed OpenDMK, you can set up a connection handler for SNMP by enabling the connection handler, and pointing OpenDJ to your installation of the OpenDMK `jdmkrt.jar` library.

```
/opt/OpenDJ-2.5.0-Xpress1# bin/dsconfig \
 set-connection-handler-prop \
 --port 4444 \
 --hostname <YOUR_HOST> \
 --bindDN "<ADMIN_CREDENTIALS>" \
 --bindPassword <ADMIN_PASSWORD> \
 --handler-name "SNMP Connection Handler" \
 --set enabled:true \
 --set opendmk-jarfile:/opt/OpenDMK-bin/lib/jdmkrt.jar \
 --trustAll \
 --no-prompt
```

By default, the SNMP Connection Handler listens on port 161 and uses port 162 for traps. On UNIX and Linux systems, only root can normally open these ports. Therefore if you install as a normal user, you might want to change the listen and trap ports:

```
/opt/OpenDJ-2.5.0-Xpress1# bin/dsconfig \
 set-connection-handler-prop \
 --port 4444 \
 --hostname <YOUR_HOST> \
 --bindDN "<ADMIN_CREDENTIALS>" \
 --bindPassword <ADMIN_PASSWORD> \
 --handler-name "SNMP Connection Handler" \
```

```
 --set listen-port:11161 \
 --set trap-port:11162 \
 --trustAll \
 --no-prompt
```

Restart the SNMP Connection Handler to take the port number changes into account.

To restart the connection handler, you disable it, then enable it again:

```
/opt/OpenDJ-2.5.0-Xpress1# bin/dsconfig \
 set-connection-handler-prop \
 --port 4444 \
 --hostname <YOUR_HOST> \
 --bindDN "<ADMIN_CREDENTIALS>" \
 --bindPassword <ADMIN_PASSWORD> \
 --handler-name "SNMP Connection Handler" \
 --set enabled:false \
 --trustAll \
 --no-prompt

/opt/OpenDJ-2.5.0-Xpress1# bin/dsconfig \
 set-connection-handler-prop \
 --port 4444 \
 --hostname <YOUR_HOST> \
 --bindDN "<ADMIN_CREDENTIALS>" \
 --bindPassword <ADMIN_PASSWORD> \
 --handler-name "SNMP Connection Handler" \
 --set enabled:true \
 --trustAll \
 --no-prompt
```

Now if following command works, you have successfully enabled SNMP on OpenDJ

```
/opt/OpenDJ-2.5.0-Xpress1# snmpwalk -v 2c -c OpenDJ@OpenDJ localhost:11161
```

JMX

OpenDJ provides Java Management eXtensions (JMX) based monitoring. A number of tools support JMX, including `jconsole` and `jvisualvm`, which are bundled with the Sun/Oracle Java platform. JMX is not configured by default. Use the `dsconfig` command to configure the JMX connection handler:

```
/opt/OpenDJ-2.5.0-Xpress1# bin/dsconfig \
 set-connection-handler-prop \
 --port 4444 \
 --hostname <YOUR_HOST> \
 --bindDN "<ADMIN_CREDENTIALS>" \
 --bindPassword <ADMIN_PASSWORD> \
 --handler-name "JMX Connection Handler" \
 --set enabled:true \
 --trustAll \
 --no-prompt
```

By default, no users have privileges to access the JMX connection. The following

command adds JMX privileges for Directory Manager:

```
/opt/OpenDJ-2.5.0-Xpress1# bin/dsconfig \
 set-root-dn-prop \
 --port 4444 \
 --hostname <YOUR_HOST> \
 --bindDN "<ADMIN_CREDENTIALS>" \
 --bindPassword <ADMIN_PASSWORD> \
 --add default-root-privilege-name:jmx-notify \
 --add default-root-privilege-name:jmx-read \
 --add default-root-privilege-name:jmx-write \
 --trustAll \
 --no-prompt
```

Now it is possible to connect via JMX using following URL:

```
service:jmx:rmi:///jndi/rmi://host:port/org.opends.server.protocols.jmx.cl
ient-unknown
```

Please use your Directory Administrator credentials and password to connect to this service. Following MBeans contain valuable information and can be found under `org.opends.server.rootDSE.cn-monitor`:

MBean	Description
cn-JVM_Memory_Usage	Monitor memory usage of the system
cn-LDAP_Connection_Handler...	Statistical breakdown of the network connection usage
cn-Disk_Space_Monitor	Monitor disk usage

2.3.9 Upgrade procedure

The OpenDJ is shipped with an integrated update routine which works as follows:

- Stop the running instance
- Backup the installation
- Download newer version
- Run upgrade script
- Rebuild indexes
- Start the upgraded instance

This can be accomplished with following commands (in this example we will upgrade a 2.4.6 installation to 2.5.0):

```
/opt/OpenDJ-2.4.6# bin/stop-ds
/opt/OpenDJ-2.4.6# cd ..
/opt# tar czf opendj.tgz OpenDJ-2.4.6
```

```
/opt# wget http://download.forgerock.org/downloads/opendj/2.5.0-
Xpress1/OpenDJ-2.5.0-Xpress1.zip
/opt# cd OpenDJ-2.4.6
/opt/OpenDJ-2.4.6# ./upgrade


Would you like to upgrade this installation to a newer version or revert
to an
older version?

    1)  Upgrade to a newer version
    2)  Revert to a previous version

Enter choice [1]: 1

Enter the name and path of the server install file (.zip): /opt/OpenDJ-
2.5.0-Xpress1.zip

Confirm Upgrade
This installation will be upgraded using the zip file
/opt/OpenDJ-2.5.0-Xpress1.zip.

    1)  Continue
    2)  Cancel

Enter choice [1]: 1

See
/var/folders/zz/zyxvpxvq6csfxvn_n0000000000000/T/opendj-upgrade-
2529653814056750585.log
for a detailed log of this operation.

Initializing Upgrade ..... Done.
Calculating Schema Customizations ..... Done.
Calculating Configuration Customizations ..... Done.
Backing Up Files ..... Done.
Upgrading Components ..... Done.
Preparing Customizations ..... Done.
Applying Configuration Customizations ..... Done.
Verifying Upgrade ..... Done.
Cleaning Up ..... Done.
Recording Upgrade History ..... Done.
See /opt/OpenDJ-2.4.6/history/log for a detailed installation history.
QuickUpgrade Completed Successfully.  The OpenDJ installation at
/opt/OpenDJ-2.4.6 has now been upgraded to version OpenDJ 2.5.0-Xpress1
(Build
ID: 20120719090339Z).

See /var/folders/zz/zyxvpxvq6csfxvn_n0000000000000/T/opendj-upgrade-
2529653814056750585.log for a detailed log of this operation.


/opt/OpenDJ-2.4.6# bin/rebuild-index \
--baseDN dc=umbrellaid,dc=org \
--index ds-sync-hist
/opt/OpenDJ-2.4.6# bin/start-ds
```

Please verify now that the directory is fully functional and that all user definitions are in-place.

3 Multiple Identity Providers

Illustration 3: Multiple Identity Providers

The Umbrella installation uses multiple Identity Providers for several reasons:

- High availability
- Load balancing
- Allow each member to have the full Umbrella database on-site

The setup allows each facility to run a physical IdP with an own LDAP server. All these IdPs form one logical IdP.

While evaluating the different clustering scenarios provided by the Shibboleth Consortium we found no existing scenario fitting the requirement to have the IdPs allocated on a WAN instead of a LAN connection. The existing scenarios all work with a replicated session state of the IdP between the clustering nodes which is unscalable due to high latency on WAN networks.

The Umbrella solution does not replicate the IdP session state but the underlying user definitions on the LDAP layer and introduces session stickiness on the DNS layer.

So the actual clustering of the IdP has no implications in the installation or configuration of the IdP, it will be configured as if it would be a standalone IdP.

Shibboleth was chosen as the SAML2 Identity Provider because of:

- Many existing and running installations in the higher education environment
- Open Source
- Proven to work
- Members of the Shibboleth Consortium are important members of the OASIS standardisation body [OASWS]

3.1 Local installation

Following prerequisites must be met to be able to run an Identity Provider:

- Apache2 webserver with AJP-proxy and SSL
- Tomcat7 servlet container
- Shibboleth Service Provider

On debian based systems following command can be used to install these components:

```
# apt-get install apache2 tomcat7 libapache2-mod-shib2
```

To enable ssl and the proxy functionality following commands can be used:

```
# a2enmod ssl
# a2enmod proxy_ajp
# a2enmod authnz_ldap
# a2enmod shib2
```

For other operating systems please refer to your operating system guidelines to install these components.

The customized and preconfigured Umbrella Identity Provider is available on github and can be installed as follows:

```
# cd /opt
/opt# git clone https://github.com/Umbrella-Commiters/UmbrellaIdP
/opt# mv UmbrellaIdP/UmbrellaIdP shibboleth-idp
/opt# rm -r UmbrellaIdP
/opt# cd shibboleth-idp
/opt/shibboleth-idp# mkdir logs
/opt/shibboleth-idp# chown tomcat7:tomcat7 logs
/opt/shibboleth-idp# cd /etc/tomcat7/Catalina/localhost/
/etc/tomcat7/Catalina/localhost# wget --no-check-certificate
https://192.33.120.67/idp.xml
```

At this point you will have an installed Identity Provider with missing x509 certificates which for security reasons are not published on github. Please get in contact with the Umbrella-Technical-Team for retrieving the missing certificates. You will get a a file called `credentials.zip`. Move this archive to the `/opt/shibboleth-idp` folder and unpack it:

```
# mv credentials.zip /opt/shibboleth-idp
# cd /opt/shibboleth-idp
/opt/shibboleth-idp# unzip credentials.zip
/opt/shibboleth-idp# rm credentials.zip
```

Also we need to add the apache directives for the IdP:

```
cp /opt/shibboleth-idp/apache/idp.conf /etc/apache2/conf.d/
```

Afterwards we need to restart tomcat and apache to reflect the changes:

```
# service tomcat7 restart
# service apache2 restart
```

Please monitor the logfile `/opt/shibboleth-idp/logs/idp-process.log` for any exceptions thrown:

```
# tail -f /opt/shibboleth-idp/logs/idp-process.log
```

3.2 Configuration

3.2.1 Core Identity Provider

The core Identity Provider downloaded at github contains all configuration changes needed to run it as an Umbrella IdP. Following an overview of what has changed compared to a vanilla Shibboleth IdP installation:

Extended `/opt/shibboleth-idp/conf/attribute-resolver.xml`

Following attributes have been added:

```
    <resolver:AttributeDefinition xsi:type="ad:Simple" id="uid"
sourceAttributeID="uid">
        <resolver:Dependency ref="myLDAP" />
        <resolver:AttributeEncoder xsi:type="enc:SAML1String"
name="urn:mace:dir:attribute-def:uid" />
        <resolver:AttributeEncoder xsi:type="enc:SAML2String"
name="urn:oid:0.9.2342.19200300.100.1.1" friendlyName="uid" />
    </resolver:AttributeDefinition>

    <resolver:AttributeDefinition xsi:type="ad:Simple" id="EAAHash"
sourceAttributeID="EAAHash">
        <resolver:Dependency ref="myLDAP" />
        <resolver:AttributeEncoder xsi:type="enc:SAML1String"
name="urn:mace:dir:attribute-def:EAAHash" />
        <resolver:AttributeEncoder xsi:type="enc:SAML2String"
name="urn:oid:1.3.6.1.4.1.9999.1.1.1" friendlyName="EAAHash" />
    </resolver:AttributeDefinition>

    <resolver:AttributeDefinition xsi:type="ad:Simple" id="EAAKey"
sourceAttributeID="EAAKey">
        <resolver:Dependency ref="myLDAP" />
        <resolver:AttributeEncoder xsi:type="enc:SAML1String"
name="urn:mace:dir:attribute-def:EAAKey" />
        <resolver:AttributeEncoder xsi:type="enc:SAML2String"
name="urn:oid:1.3.6.1.4.1.9999.1.1.3" friendlyName="EAAKey" />
    </resolver:AttributeDefinition>

    <resolver:AttributeDefinition xsi:type="Script"
xmlns="urn:mace:shibboleth:2.0:resolver:ad"
                                  dependencyOnly="true"
                                  id="x500Principal">
        <Script><![CDATA[

importPackage(Packages.edu.internet2.middleware.shibboleth.common.attribut
e.provider);
            importPackage(Packages.javax.security.auth.x500);

            x500Principal = new BasicAttribute("x500Principal");
            if (requestContext.getUserSession())
            {
                subject = requestContext.getUserSession().getSubject();
                if (subject != null) {

x500Principal.getValues().addAll(subject.getPrincipals(X500Principal("").g
```

```
etClass()));
                }
            }
        ]]></Script>

    </resolver:AttributeDefinition>

    <!-- Provides the subjectAltNames of type rfc822Name from the
certificate as attribute -->
    <resolver:AttributeDefinition xsi:type="Script"

xmlns="urn:mace:shibboleth:2.0:resolver:ad"
                                  dependencyOnly="true"
                                  id="x500SubjectAltNameEMailPrincipal">
        <Script><![CDATA[

importPackage(Packages.edu.internet2.middleware.shibboleth.common.attribut
e.provider);
            importPackage(Packages.ch.SWITCH.aai.idp.x509.principals);

            x500SubjectAltNameEMailPrincipal = new
BasicAttribute("x500SubjectAltNameEMailPrincipal");
            if (requestContext.getUserSession())
            {
                subject = requestContext.getUserSession().getSubject();
                if (subject != null) {

x500SubjectAltNameEMailPrincipal.getValues().addAll(subject.getPrincipals(
EMailPrincipal("").getClass()));
                }
            }
        ]]></Script>
    </resolver:AttributeDefinition>
    <!-- Name Identifier related attributes -->
    <resolver:AttributeDefinition id="transientId"
xsi:type="ad:TransientId">
        <resolver:AttributeEncoder
xsi:type="enc:SAML1StringNameIdentifier"
nameFormat="urn:mace:shibboleth:1.0:nameIdentifier"/>
        <resolver:AttributeEncoder xsi:type="enc:SAML2StringNameID"
nameFormat="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"/>
    </resolver:AttributeDefinition>

    <!-- Jean-Francois EAAHash as persistent NameID
     Attribute definition that expects to get the 'EAAHash' attribute from
the ldap connector
     defined as its dependency and encode it as a SAML 2 name identifier.
    -->
    <resolver:AttributeDefinition xsi:type="Simple"
xmlns="urn:mace:shibboleth:2.0:resolver:ad"
                              id="EAAHashAsNameID"
                              sourceAttributeID="EAAHash">
      <resolver:Dependency ref="myLDAP" />
      <!-- Encoder that transforms the attribute into a SAML2 NameID -->
      <resolver:AttributeEncoder xsi:type="SAML2StringNameID"
xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
nameFormat="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent" />
```

```
    </resolver:AttributeDefinition>
```

Following LDAP DataConnector was added:

```
    <resolver:DataConnector id="myLDAP" xsi:type="dc:LDAPDirectory"
        ldapURL="ldap://localhost"
        baseDN="ou=people,dc=umbrellaid,dc=org"
        principal="<ADMIN_CREDENTIALS>"
        principalCredential="<ADMIN_PASSWORD>">
      <resolver:Dependency ref="x500Principal" />
      <resolver:Dependency ref="x500SubjectAltNameEMailPrincipal" />
        <!-- Example for using X.509 in conjuntion with UsernamePassword
Login Handler
             Using CN or one of the provided E-Mail addresses -->
        <dc:FilterTemplate><![CDATA[
            #if( !$x500Principal.Empty )
                #set( $dn = $x500Principal.get(0).name.split(",") )
                #foreach( $item in $dn )
                    #if( $item.startsWith("CN=") )
                        #set( $cn = $item.substring(3) )
                    #end
                #end

                (| #foreach($mail in $x500SubjectAltNameEMailPrincipal)
(mail=$mail) #end #if($cn) (cn=$cn) #end)

            #else
                (uid=$requestContext.principalName)
            #end
        ]]></dc:FilterTemplate>

    </resolver:DataConnector>
```

Extended `/opt/shibboleth-idp/conf/attribute-filter.xml`

Add following AttributeFilterPolicy to release all attributes to all parties:

```
  <afp:AttributeFilterPolicy id="releaseTransientIdToAnyone">
        <afp:PolicyRequirementRule xsi:type="basic:ANY"/>
        <afp:AttributeRule attributeID="transientId">
            <afp:PermitValueRule xsi:type="basic:ANY"/>
        </afp:AttributeRule>
        <afp:AttributeRule attributeID="EAAHashAsNameID">
            <afp:PermitValueRule xsi:type="basic:ANY"/>
        </afp:AttributeRule>
        <afp:AttributeRule attributeID="uid">
            <afp:PermitValueRule xsi:type="basic:ANY" />
        </afp:AttributeRule>
        <afp:AttributeRule attributeID="EAAHash">
            <afp:PermitValueRule xsi:type="basic:ANY" />
        </afp:AttributeRule>
        <afp:AttributeRule attributeID="EAAKey">
            <afp:PermitValueRule xsi:type="basic:ANY" />
        </afp:AttributeRule>
    </afp:AttributeFilterPolicy>
```

Extended /opt/shibboleth-idp/conf/relying-party.xml

Add following DefaultRelyingParty definition. This will include the ECP capabilities as well:

```
<rp:DefaultRelyingParty provider="https://umbrellaid.org/idp/shibboleth"
defaultSigningCredentialRef="IdPCredential">

   <rp:ProfileConfiguration xsi:type="saml:ShibbolethSSOProfile"
includeAttributeStatement="false" assertionLifetime="PT5M"
signResponses="conditional" signAssertions="never" />

   <rp:ProfileConfiguration xsi:type="saml:SAML1AttributeQueryProfile"
assertionLifetime="PT5M" signResponses="conditional"
signAssertions="never" />

   <rp:ProfileConfiguration xsi:type="saml:SAML1ArtifactResolutionProfile"
signResponses="conditional" signAssertions="never" />

   <rp:ProfileConfiguration xsi:type="saml:SAML2SSOProfile"
includeAttributeStatement="true" assertionLifetime="PT5M"
assertionProxyCount="0" signResponses="never" signAssertions="always"
encryptAssertions="conditional" encryptNameIds="never" />

   <rp:ProfileConfiguration xsi:type="saml:SAML2ECPProfile"
includeAttributeStatement="true" assertionLifetime="PT5M"
assertionProxyCount="0" signResponses="never" signAssertions="always"
encryptAssertions="never" encryptNameIds="never" />

   <rp:ProfileConfiguration xsi:type="saml:SAML2AttributeQueryProfile"
assertionLifetime="PT5M" assertionProxyCount="0"
signResponses="conditional" signAssertions="never"
encryptAssertions="conditional" encryptNameIds="never" />

   <rp:ProfileConfiguration xsi:type="saml:SAML2ArtifactResolutionProfile"
signResponses="never" signAssertions="always"
encryptAssertions="conditional" encryptNameIds="never" />

   <rp:ProfileConfiguration xsi:type="saml:SAML2LogoutRequestProfile"
signResponses="always" signAssertions="never" encryptAssertions="never"
encryptNameIds="never" backChannelConnectionPoolTimeout="2000"
backChannelConnectionTimeout="2000" backChannelResponseTimeout="5000" />

</rp:DefaultRelyingParty>
```

Add following metadata definition:

```
<metadata:MetadataProvider id="ShibbolethMetadata"
xsi:type="metadata:ChainingMetadataProvider">
        <metadata:MetadataProvider id="IdPMD"
xsi:type="metadata:FilesystemMetadataProvider"
metadataFile="/opt/shibboleth-idp/metadata/idp-metadata.xml"
maxRefreshDelay="P1D" />
</metadata:MetadataProvider>
```

Extended /opt/shibboleth-idp/conf/handler.xml

3.2.2 Service Provider of IdP

The IdP Service Provider is needed for the ECP extension as well as for the Umbrella Webapp. The installation of the apache module is covered in chapter “3.1 Local installation”. Following configuration changes must be done:

```
# cd /etc/shibboleth
/etc/shibboleth# mv shibboleth2.xml shibboleth2.xml.dist
/etc/shibboleth# mv attribute-map.xml attribute-map.xml.dist

/etc/shibboleth# wget https://github.com/Umbrella-
Commiters/UmbrellaFrontend/raw/master/config/shibboleth/shibboleth2.xml

/etc/shibboleth# wget https://github.com/Umbrella-
Commiters/UmbrellaFrontend/raw/master/config/shibboleth/attribute-map.xml

/etc/shibboleth# wget https://github.com/Umbrella-
Commiters/UmbrellaFrontend/raw/master/config/shibboleth/sp-cert.pem
```

Additionaly you need to contact the Umbrella Technical Team to retrieve the private key for the Service Provider and place it under `/etc/shibboleth` as well.

Restart both `apache2` and `shibd` afterwards:

```
# service apache2 restart
# service shibd restart
```

This service provider currently covers:

- the ECP extension which is configured in the file `/etc/apache2/conf.d/idp.conf` downloaded during the local installation of the IdP covered in chapter 3.1.
- the Umbrella webapp which is configured in the file `/etc/apache2/conf.d/euu.conf` downloaded during the local installation of the webapp covered in chapter 4.1.

3.2.3 SLO Module

The “Single Logout” functionality is part of the SAML specification [OASWS] but it is not implemented within a vanilla Shibboleth Identity Provider.

The Hungarian NREN NIIF provides a re-packaged Shibboleth Identity Provider [NIIFSLO] which is used for the Umbrella IdP to provide it with the SLO functionality.

The installation of this patched IdP is very similar to a vanilla installation:

```
/tmp# wget
http://software.niif.hu/maven2/edu/internet2/middleware/shibboleth-
identityprovider/2.3.8-slo10/shibboleth-identityprovider-2.3.8-slo10-
bin.tar.gz
```

```
/tmp# tar -zxvf shibboleth-identityprovider-2.3.8-slo10-bin.tar.gz
/tmp# cd shibboleth-identityprovider-2.3.8-slo10
/tmp/shibboleth-identityprovider-2.3.8-slo10# ./install.sh

install:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Be sure you have read the installation/upgrade instructions on the
Shibboleth website before proceeding.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Where should the Shibboleth Identity Provider software be installed?
[/opt/shibboleth-idp]
What is the fully qualified hostname of the Shibboleth Identity Provider
server? [idp.example.org]
idp.umbrellaid.org
A keystore is about to be generated for you. Please enter a password that
will be used to protect it.
Changeasap

Updating property file: /Users/bjoern/Downloads/shibboleth-
identityprovider-2.3.8-slo10/src/installer/resources/install.properties
Created dir: /opt/shibboleth-idp
Created dir: /opt/shibboleth-idp/bin
Created dir: /opt/shibboleth-idp/conf
Created dir: /opt/shibboleth-idp/credentials
Created dir: /opt/shibboleth-idp/lib
Created dir: /opt/shibboleth-idp/lib/endorsed
Created dir: /opt/shibboleth-idp/logs
Created dir: /opt/shibboleth-idp/metadata
Created dir: /opt/shibboleth-idp/war
Generating signing and encryption key, certificate, and keystore.
Copying 5 files to /opt/shibboleth-idp/bin
Copying 9 files to /opt/shibboleth-idp/conf
Copying 1 file to /opt/shibboleth-idp/metadata
Copying 51 files to /opt/shibboleth-idp/lib
Copying 5 files to /opt/shibboleth-idp/lib/endorsed
Copying 1 file to /Users/bjoern/Downloads/shibboleth-identityprovider-
2.3.8-slo10/src/installer
Building war: /Users/bjoern/Downloads/shibboleth-identityprovider-2.3.8-
slo10/src/installer/idp.war
Copying 1 file to /opt/shibboleth-idp/war
Deleting: /Users/bjoern/Downloads/shibboleth-identityprovider-2.3.8-
slo10/src/installer/web.xml
Deleting: /Users/bjoern/Downloads/shibboleth-identityprovider-2.3.8-
slo10/src/installer/idp.war

BUILD SUCCESSFUL
```

After this procedure we have an installed but not configured Identity Provider. This installation is the basis for the provided Umbrella IdP at github and should be used to create upgrades of the IdP. The version 2.3.8 will be the last version where a complete IdP must be downloaded. From version 2.4.0 on this functionality can be provided as an external module and it's planned to be released for version 3.0.0.

3.2.4 ECP Extension

At the time of writing the version of the identity provider already comes with an ECP extension shipped, so it must just be activated by configuration, although the preconfigured IdP already contains all changes.

Following bits form the configuration:

- Apache configuration
- ProfileHandler
- ProfileConfiguration
- Metadata

Apache configuration

The apache configuration is delivered in the file `/opt/shibboleth-idp/apache/idp.conf` and must be copied into the apache configuration. The exact command is described in chapter “3.1 Local installation”.

Two Locations are described in here:

Location	Description
/secure	Used for initiating the login process
/idp/profile/SAML2/SOAP/ECP	The IdP endpoint actually executing the login against the Umbrella user directory.

ProfileHandler

Following `ProfileHandler` must be present in the file `/opt/shibboleth-idp/conf/handler.xml`:

```
<ph:ProfileHandler xsi:type="ph:SAML2ECP"
        inboundBinding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"

outboundBindingEnumeration="urn:oasis:names:tc:SAML:2.0:bindings:SOAP">
        <ph:RequestPath>/SAML2/SOAP/ECP</ph:RequestPath>
</ph:ProfileHandler>
```

ProfileConfiguration

Following `ProfileConfiguration` must be present in the file `/opt/shibboleth-idp/conf/relying-party.xml`:

```
<rp:ProfileConfiguration xsi:type="saml:SAML2ECPProfile"
includeAttributeStatement="true"
        assertionLifetime="PT5M" assertionProxyCount="0"
        signResponses="never" signAssertions="always"
        encryptAssertions="never" encryptNameIds="never"/>
```

Metadata

Each Umbrella Identity Provider needs following stanza in its metadata definition to

support ECP:

```
<SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
Location="https://umbrellaid.org/idp/profile/SAML2/SOAP/ECP" />
```

Each Umbrella Service Provider needs following stanza in its metadata definition to support ECP (replace <YOUR_SP_URL> with your actual SP URL, e.g. https://umbrellaid.org):

```
<AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:PAOS"
Location="<YOUR_SP_URL>/Shibboleth.sso/SAML2/ECP" index="4" />
```

3.2.5 EduGain Bridge

3.2.6 Moonshot Extension

3.3 Operations

3.3.1 Define user metadata

User metadata is defined as attributes containing certain information about a specific user, e.g. a username. This information must be storable, retrievable, releasable and consumable:

- The information is stored in the directory
- The information is retrieved during a login
- The information is released after a successful login via SAML2
- The information is consumed by a Service Provider

Following parts must be adjusted if user metadata must be changed:

- Change the schema configuration on the Identity Provider in the file: `<LDAP_HOME>/config/schema/99-user.ldif`
- Change the Attribute Definitions on the Identity Provider in the file: `<SHIB_HOME>/conf/attribute-resolver.xml`
- Change the Attribute Rules on the Identity Provider in the file: `<SHIB_HOME>/conf/attribute-filter.xml`
- Change the Attributes on each Service Provider in the file: /etc/shibboleth/`attribute-map.xml`

3.3.2 Define host metadata

Host metadata describes entities participating in a federation. These can be of following types:

- IDPSSODescriptor
- SPSSODescriptor

Both types define for their corresponding entity information like endpoints, public-keys,

contact information, etc.

The IdP definition should not change. Exceptions are upgrades to a newer SAML protocol, introductions of new endpoints or changes of the product.

SP metadata definitions must be added when new members join the Umbrella federation. Specific instructions are described in chapter “6 Service Provider Setup”.

The SP metadata definition must be added to each Identity Provider metadata definition in the file: `<SHIB_HOME>/metadata/idp-metadata.xml`

3.3.3 Define LoginHandler

3.3.4 Local backup

As the Identity Provider has no database or directory, its backup is relatively simple. We just need to backup the installation under `/opt/shibboleth-idp` on the filesystem.

3.3.5 Logfiles

Logfiles are an important source of information if we need to debug a specific behavior of the server. Following logfiles are available:

Path	Description
logs/idp-process.log	Contains messages logged during the normal operation of the IdP. This log is meant to be human readable and contains messages that indicate what the IdP is currently doing, encountered errors, warning messages that may indicate potential problems, etc.
logs/idp-access.log	Contains a log entry for each time the IdP is accessed, whether information was ever sent back or not. These messages include request time, remote host making the request, server host name and port, and the request path. This log is written in the machine parsable format requestTime|remoteHost|serverHost|serverPort| requestPath|.
logs/idp-audit.log	Contains a log entry for each time the IdP sends data to a relying party. These messages include the audit event time, IdP and relying party IDs, request and response binding, communication profile ID, request and response ID, principal name, authentication method, and released attribute of the current user. This log is written in the machine parsable format auditEventTime|requestBinding|requestId| relyingPartyId|messageProfileId|assertingPartyId|responseBinding| responseId|principalName|authNMethod| releasedAttributeId1,releasedAttributeId2,|nameIdentifier| assertion1ID,assertion2ID,|

The logging configuration for the IdP is located at $IDP_HOME/conf/logging.xml. This file is checked for changes every 10 minutes by default and is reloaded if changes have been made. This means a deployer can keep the logging level at WARN until a problem occurs and then change the logging to DEBUG to get more information if the problem persists, all

without restarting the IdP.

The following, coarse grained, loggers provide useful information in most situations:

Category	Description
Shibboleth-Access	The logger to which shibboleth access messages (think HTTP access logs) are written
Shibboleth-Audit	The logger to which shibboleth audit messages are written
PROTOCOL_MESSAGE	The logger to which incoming and outgoing XML protocol messages are logged
org.opensaml	Messages related only to receiving, parsing, evaluating security of, producing, and encoding SAML messages. Note, this produces a lot of log messages, especially at IdP startup.
org.opensaml.saml2.metadata.provider	Information regarding metadata loading, refreshing, and querying.
edu.internet2.middleware.shibboleth	Messages related to all the non-SAML message parsing/encoding work; profile handling, authentication, attribute resolution and filtering
edu.internet2.middleware.shibboleth.idp.authn	IdP messages related only to authentication
edu.internet2.middleware.shibboleth.common.relyingparty	IdP messages related to relying party configuration in use
edu.internet2.middleware.shibboleth.common.attribute	IdP messages related only to attribute resolution and filtering

The logback system defines 5 logging levels:

- TRACE
- DEBUG
- INFO
- WARN
- ERROR

As you progress from the highest level (ERROR) to the lowest level (TRACE) the amount of information logged increases (dramatically so on the DEBUG and TRACE levels). Each level also logs all messages of the levels above it. For example, INFO also logs WARN and ERROR messages.

3.3.6 Monitoring

TBD. Cacti Scripts for SNMP monitoring.

3.3.7 Upgrade procedure

Upgrading an IdP consists of following steps:

1. Check-Out the UmbrellaIdP project from github
2. Download the new version of the Shibboleth IdP
3. Configure the new version of the IdP on a test server and get it running and make sure everything works as expected.
4. Check-In the UmbrellaIdP project back to github with the new identity provider.
5. Remove your node from GeoDNS.
6. Configure the new version on the live server and make sure everything works.
7. Add your node back to GeoDNS
8. Remove all other nodes.
9. For each other node repeat steps 1,6,7.

4 Umbrella Webapp

4.1 Local installations

It is assumed that you have installed Maven and wget as Maven is used for dependency management.

Following commands will check-out the source from github and build the whole project:

```
# cd /tmp
/tmp# mkdir build
/tmp# cd build
/tmp/build# wget http://kaptcha.googlecode.com/files/kaptcha-2.3.2.zip
/tmp/build# unzip kaptcha-2.3.2.zip kaptcha-2.3.2.jar
/tmp/build# mvn install:install-file -DgroupId=com.google.code.kaptcha
-DartifactId=kaptcha -Dversion=2.3 -Dpackaging=jar -Dfile=kaptcha-
2.3.2.jar
/tmp/build# wget
https://github.com/flowedback/UmbrellaFrontend/archive/master.zip
/tmp/build# unzip master.zip
/tmp/build# cd UmbrellaFrontend-master
/tmp/build/UmbrellaFrontend-master# wget
https://raw.github.com/spaetow/UmbrellaFrontend/master/pom.xml -O pom.xml
/tmp/build/UmbrellaFrontend-master# mvn clean
```

Now we need to adjust the configuration to match our local settings. This is done in the file `src/main/resources/eu/eurofel/messages.properties`:

```
notification.enabled=true

mail.dateformat=dd.MM.yyyy
mail.from=no-reply@umbrellaid.org
mail.port=<mail server port> (default: 25)
mail.host=<mail server name> (if you installed postfix: localhost)
mail.username=<username to access mail server> (if required)
mail.password=<password for username> (if required)
application.context.path=/var/lib/tomcat7/webapps/euu/WEB-
INF/applicationContext.xml
eaa.url=https://umbrellaid.org/
eaa.path.file=/var/www/includes
eaa.path.www=https://umbrellaid.org/includes
```

Now we need to package the application and copy it to the webapps folder of tomcat:

```
/tmp/build/UmbrellaFrontend-master# mvn package
```

```
/tmp/build/UmbrellaFrontend-master# cp target/euu.war
/var/lib/tomcat7/webapps/
```

If tomcat isn't set to auto-deploy you will need to restart tomcat

```
# service tomcat7 restart
```

4.2 Configuration

The main configuration is located in the file `src/main/resources/eu/eurofel/messages.properties`:

Name	Description
notification.enabled	Do you want to send notifications: true/false
mail.dateformat	
mail.from	The sender address for outgoing email
mail.port	The port of the mail server (usually 25)
mail.host	The hostname or IP address of the mail server
mail.username	Username to authenticate against the mail server (if needed)
mail.password	Password of the mail user (if needed)
application.context.path	Path to the applicationContext.xml Spring configuration file
eaa.url	The URL used for the application. Is used when sending email
eaa.path.file	Path in the filesystem to the typo3 navigation export
eaa.path.www	Web URL to access the typo3 navigation export

The directory configuration is externalised and is located in the file: `src/main/resources/eaa.properties:`

Name	Description
eaa.initial_context_factory	The initial context factory used to connect to the directory. This usually is: com.sun.jndi.ldap.LdapCtxFactory
eaa.provider_url	The URL used to connect to the directory. If the directory runs on the same host this would be: ldap://localhost:389
eaa.security_authentication	The authentication type. In most cases this is: simple
eaa.security_principal	The distinguished name used to connect to the directory
eaa.security_credentials	The directory users password
eaa.people_root	The DN where people definitions reside. This currently is: ou=people,dc=umbrellaid,dc=org

4.3 Operations

4.3.1 Local Backup

As this web application doesn't persist anything, the backup is fairly straightforward. Just backup the web application itself and it's servlet containers logfiles.

In the case of tomcat this would include following files and folders recursively:

- /var/lib/tomcat7/webapps/euu
- /var/lib/tomcat7/webapps/euu.war
- /var/log/tomcat7

4.3.2 Logfiles

The Umbrella webapp uses the default tomcat logfiles. This usually include following files:

- /var/log/tomcat7/catalina.out
- /var/log/tomcat7/localhost.YYYY-MM-DD.log

4.3.3 Monitoring

To monitor the application you should monitor your servlet container. Please refer to the specific documentation of the servlet container used.

4.3.4 Upgrade procedure

To upgrade the application please refer to chapter “4.1 Local installation” and repeat the steps propagated

5 GeoDNS

Geo DNS

GeoDNS is a regular DNS service but takes the geographic location of a client into account and can deliver a specific IP address for the Umbrella service based on the geographic information.

This functionality is used to balance the load on the Umbrella system to different hosts.

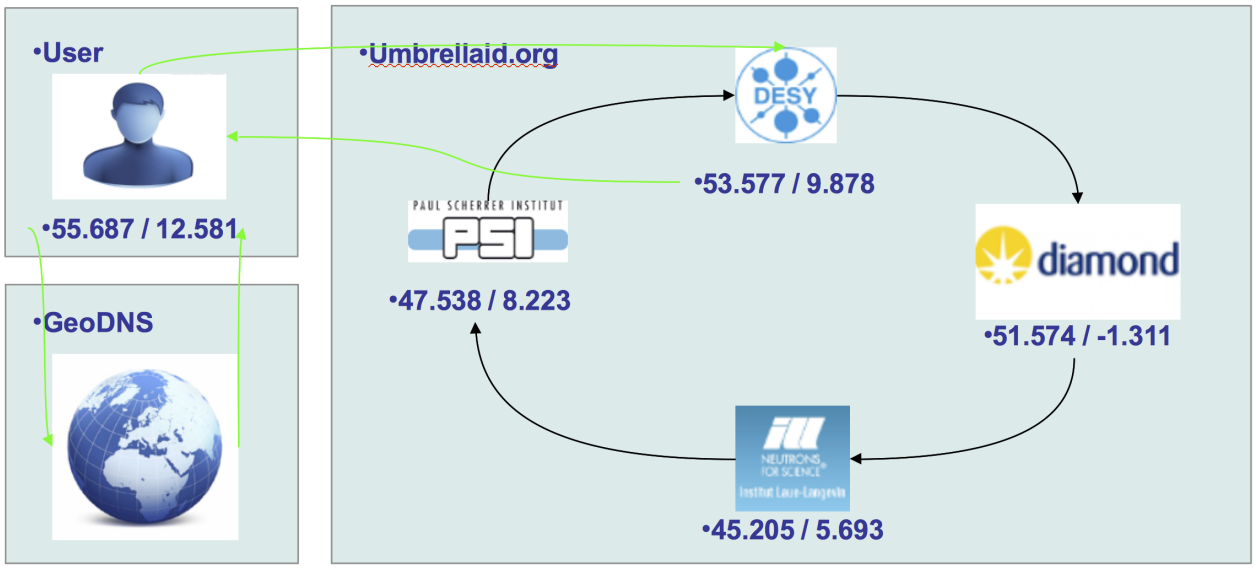


Illustration 4: GeoDNS resolution and ring

The user will always be redirected to the closest IdP.

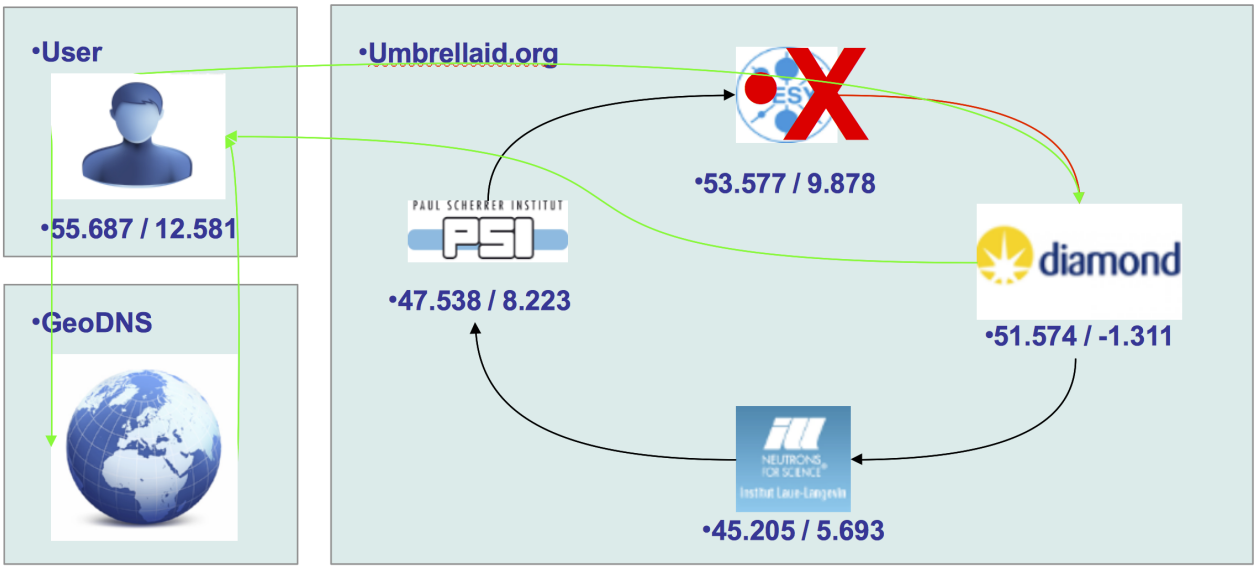


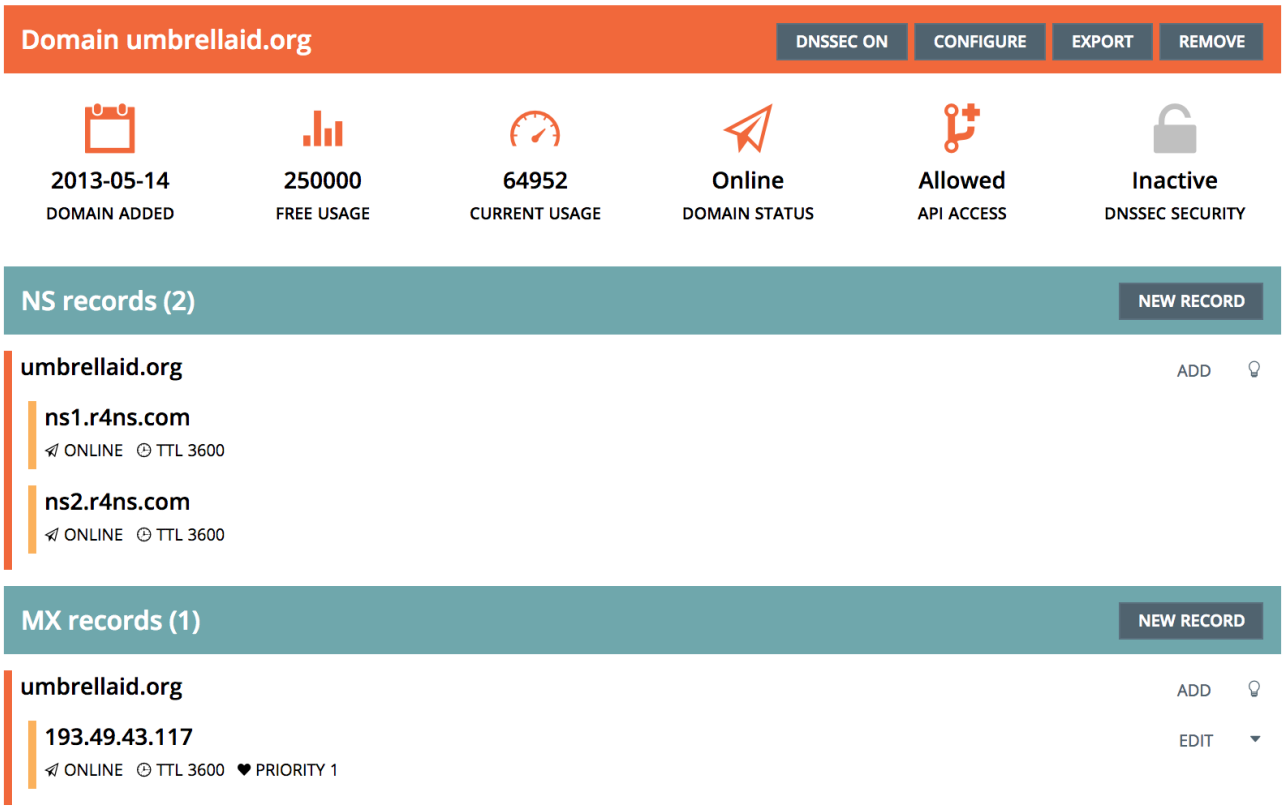
Illustration 5: GeoDNS failover

If a host fails to respond it will be deactivated and all request to this system will be forwarded to the next host in the failover ring of GeoDNS.

5.1 Basic Configuration

The service used for Umbrella is a commercial one called rage4.com. It allows a certain amount of requests for free and then the costs are based on the actual amount of requests processed.

The service offers a web interface to configure it. You can access it via <https://rage4.com> and looks as follows:



It allows you to do all standard DNS operations in a user friendly way.

5.2 Extended GeoDNS with failover

To extend a DNS record to be part of GeoDNS and failover we need to edit an existing A record and navigate to the specific tabs “GeoDNS configuration” and “Failover configuration”.

Following information is needed prior to configuration:

Name	Description
Server latitude	The latitude of the specific Umbrella node in decimal degree format
Server longitude	The longitude of the specific Umbrella node in decimal degree format
Next failover node	The IP address of the next failover host.

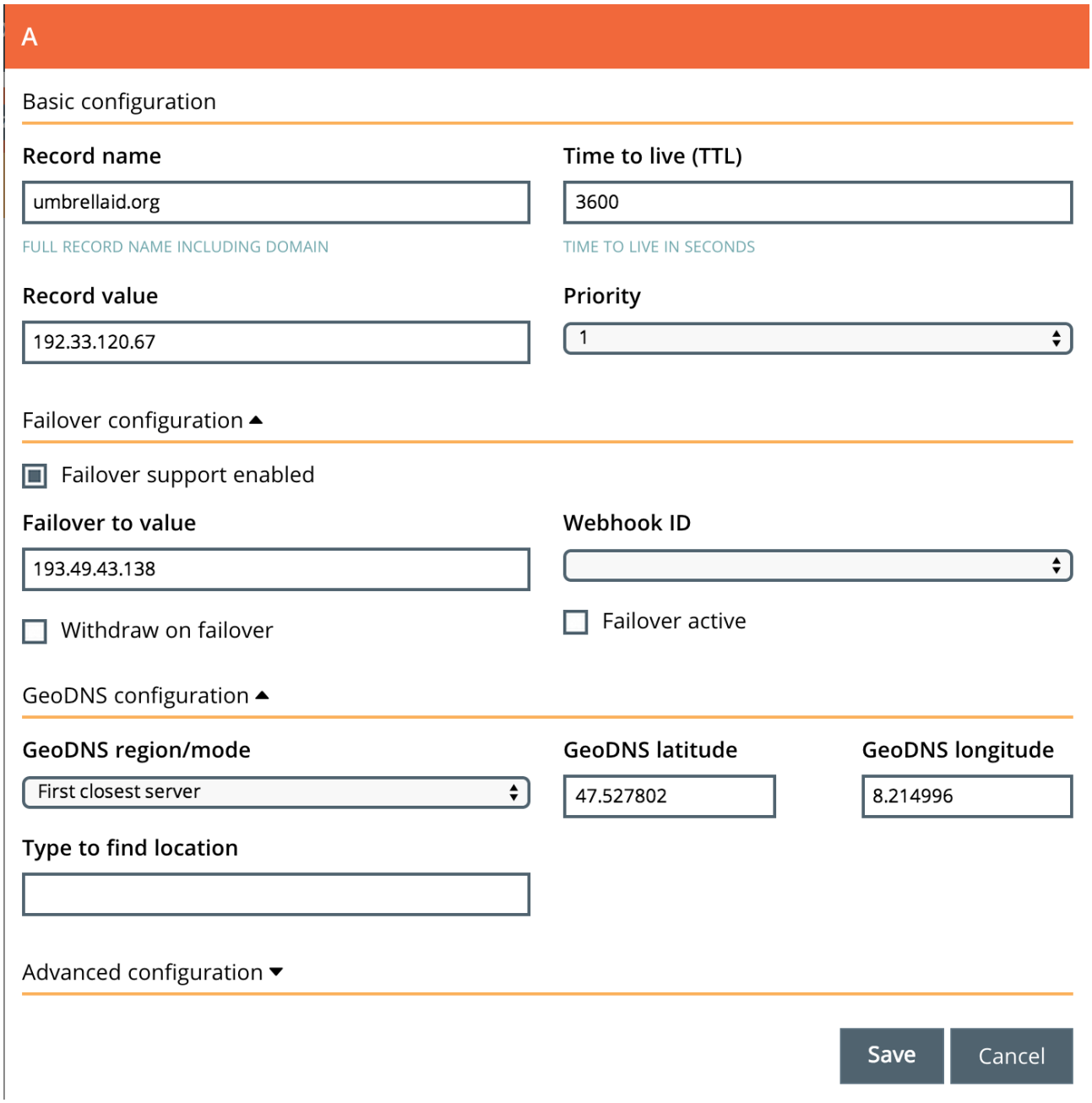
To determine the “Next failover node” you have to fit the new node into the existing infrastructure. We propose a ring system as shown in the Illustration “GeoDNS resolution and ring” which brings us to following pattern: HOST(FAILOVER) -> HOST(FAILOVER) -> HOST(FAILOVER).

Example: HOST A and HOST B exist and we want to add HOST C

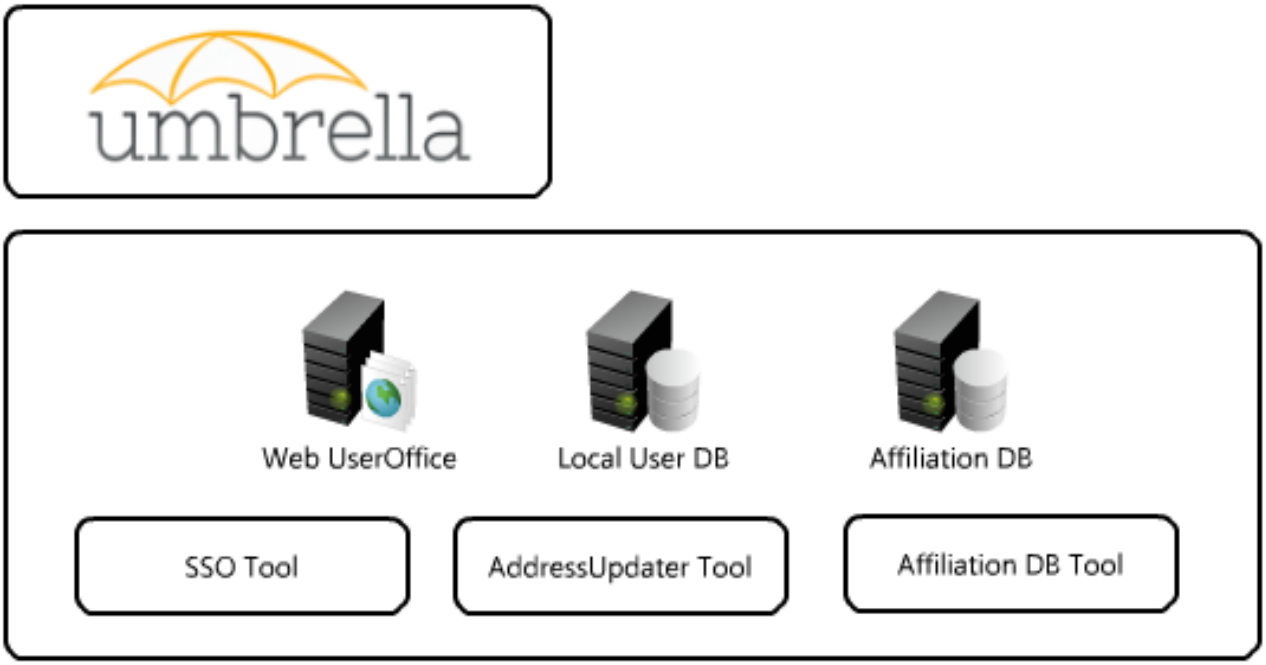
Existing config:HOST A(HOST B) -> HOST B(HOST A)

New config: HOST A(HOST B) -> HOST B(HOST C) -> HOST C(HOST A)

Following a screenshot of the configuration in the web interface of rage4.com



6 Service Provider Setup

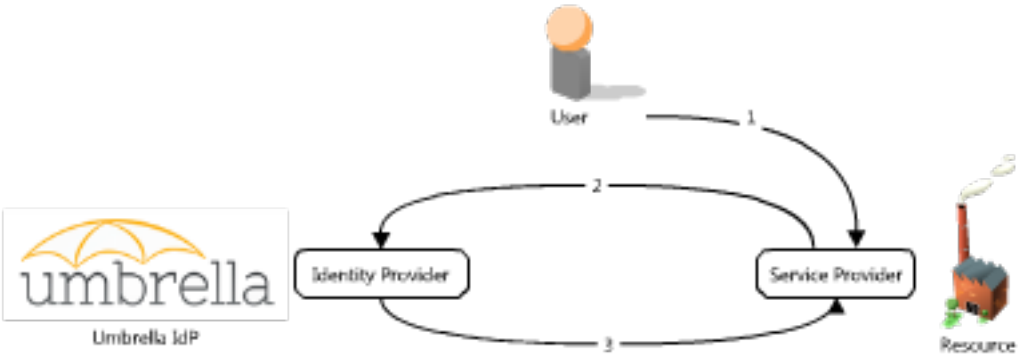


A SAML2 service provider is needed for all services which would like to participate to enable the communication between user, identity provider and service provider. The service provider is plugged-in to a web server and can protect specific web resources.

There are different implementations of SAML2 service providers:

- Shibboleth SP
- SimpleSAMLphp
- ADFS Relying Party

All of them can be used to participate in the Umbrella federation. They work in following way:



1. The user requests access to a protected resource.

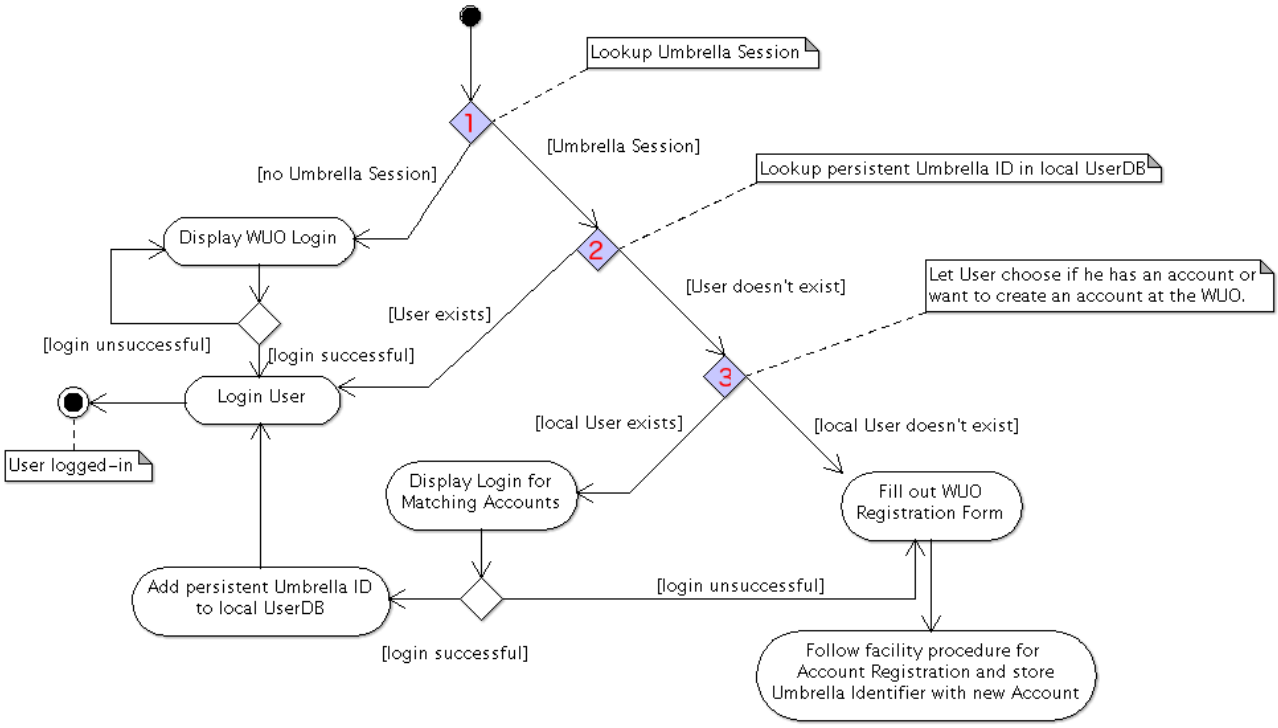
2. The service provider of the protected resource redirects the user to the Umbrella identity provider requesting a login.
3. If login is successful or a session already exists the user is being redirected back to the protected resource allowing access to the resource.

6.1 Concept

To be able to participate at the Umbrella the WUOs must install a SAML2 service provider software, e.g. mod_shib2 (Apache) or SimpleSAMLphp (PHP), to be able to receive SAML2 attributes. The metadata of this service provider must then be installed in the Umbrella IdP.

Since the page should be accessible when no EAA-Session exists, the SAML2isPassive value must be set.

To enable the hybrid characteristics of the authentication system both authenticating at the WUO and the Umbrella must be possible. This is accomplished through a multi-phase check at the WUO itself:



It is presumed that following procedures are already installed at the WUOs as this integration builds upon them:

- Login Form
- Registration Form

The Login Form is shown to incoming Umbrella-Users who have a WUOAccount to let them match the two accounts.

The Create Account Form is shown to incoming Umbrella-Users who don't have a WUO-

Account to let them create a new account according to the guidelines of the WUO.

1. Umbrella-Session check: Check if there is an active existing Umbrella-Session. If there is no valid Umbrella-Session, continue with the normal application code. Else forward the user to User Check.
2. User Check: Query the database for the incoming Umbrella-Hash. If there is a matching user, log him in. Else forward the user to user matching.
3. User Matching: Let the user choose to match his Umbrella-Account with an existing WUO-account or let him create a new WUO-account.

6.2 Local installation

Following commands are needed to install the Shibboleth service provider and Umbrella configuration:

```
# apt-get install libapache2-mod-shib2
# cd /etc/shibboleth/
/etc/shibboleth/# wget https://github.com/Umbrella-
Commiters/UmbrellaFrontend/raw/master/config/shibboleth/attribute-map.xml
```

6.3 Configuration

The service provider needs to be configured to be part of the Umbrella federation. This is done in the file /etc/shibboleth/shibboleth2.xml:

```
# vi /etc/shibboleth/shibboleth2.xml
```

Following parts must be adapted:

- ApplicationDefaults:

```
<ApplicationDefaults entityID="<YOUR_ENTITY_ID>" REMOTE_USER="uid">
```

- SSO:

```
<SSO entityID="https://umbrellaid.org/idp/shibboleth">
      SAML2 SAML1
</SSO>
```

- MetadataProvider:

```
<MetadataProvider type="XML" uri="http://umbrellaid.org/metadata/idp-
metadata.xml" backingFilePath="federation-metadata.xml"
reloadInterval="7200" />
```

6.4 Adaption of the local web application

Your local web application must be adapted to the Umbrella workflow:

6.4.1 Umbrella-Session check

To find out if a user has a session, we need to read HTTP server headers for the attribute EAAHash. Following a few examples:

Language	Construct
Java	`httpServletRequest.getHeader(“EAAHash”);`
PHP	`$_SERVER[“EAAHash”];`

6.4.2 User check

The User Check consists basically of an extension to the SQL SELECT statement which besides querying for username and password also queries for the incoming EAAHash.

Following code snippet (red and bold) can be used to enhance this generic user query:

```
SELECT
   USERNAME
FROM
   USERS
WHERE
   (USERNAME='user' AND PASSWORD='changeit')
OR
   (EAAHASH='f5bba3c6-6240-4ccf-8048-13dbb3405192')
```

6.4.3 User matching

User Matching is necessary if there is an incoming Umbrella-Session but no WUO user registered with it. It's important to use the existing “Login” and “Create User” procedures already installed at the facilities, so that no new procedures must be installed and approved.

There is a chance that the user already has an existing account at the WUO and that the user wants to bind those accounts together – then a WUO login is performed and if successful, the found user tuple is enhanced in the USERS table with the incoming EAAHash.

```
UPDATE
   USERS
SET
   EAAHASH='f5bba3c6-6240-4ccf-8048-13dbb3405192'
WHERE
   USERID='foundID'
```

If the user has no existing account the WUO's user creation process is used to create a new WUO user. The EAAHash must then be appended to the created user.

```
INSERT INTO USERS
   (NAME,…,EAAHASH)
VALUES
   ('Muster',…,'f5bba3c6-6240-4ccf-8048-13dbb3405192')
```

6.5 Umbrella Tools

Several tools have been created for the Umbrella system to make live easier for users. Following a list of these tools and the steps needed to integrate them:

6.5.1 AddressUpdater

The AddressUpdater is a tool which can retrieve user information from a facility where the user is registered at and display it at the Umbrella website so that the user can mutate his information and submit it to all facilities.

Information Retriever

To retrieve the information, the WUO must enable a UserInformation-Endpoint, where a user logged in to the Umbrella can get a list of his information registered at this specific facility. As this usually is a cross-domain request, JSONP must be used here.

Following a code snippet in PHP which explains the functionality:

```
// retrieve the Umbrella ID from the Headers
$shibhash = $_SERVER["EAAHash"];

// make sure it is not empty
if($shibhash <> ""){

   // query for the attributes for the specific user
   $result=$db->Execute("SELECT USERNAME,PASSWORD,USERID,TITLE||' '||
FIRSTNAME||' '||MIDDLENAME||' '||LASTNAME AS
FULLNAME,EMAIL,STATUS,FIRSTNAME,MIDDLENAME,LASTNAME,PHONE,TITLE,SEX FROM
USERS WHERE EAAHASH=:p1",
   array("p1" => $shibhash) );

   // retrieve the attributes from the database
   $userid=$result->fields[2];
   $fullname=$result->fields[3];
   $useremail=$result->fields[4];
   $firstname=$result->fields[6];
   $middlename=$result->fields[7];
   $lastname=$result->fields[8];
   $phone=$result->fields[9];
   $title=$result->fields[10];
   $sex=$result->fields[11];
   // display the attributes as JSONP
   echo
   $_GET["jsonp"] ."({\"Userid\": \"$userid\",\"Fullname\": \"$fullname\",
\"Email\": \"$useremail\", \"Firstname\": \"$firstname\", \"Middlename\":
\"$middlename\", \"Lastname\": \"$lastname\", \"Phone\": \"$phone\", \"Tit
```

```
le\": \"$title\", \"Gender\": \"$sex\"})";
}
```

If a client calls this endpoint he will receive a JSONP answer in following format:

```
asuidfgaiiq38zrfwhsudf({
   "Userid": "2",
   "Fullname": "Mr. Bjoern Erik Abt",
   "Email": "bjoern.abt@psi.ch",
   "Firstname": "Bjoern",
   "Middlename": "Erik",
   "Lastname": "Abt",
   "Phone": "0041563103509",
   "Title": "Mr.",
   "Gender": "M"
})
```

6.5.2 AttributeUpdater

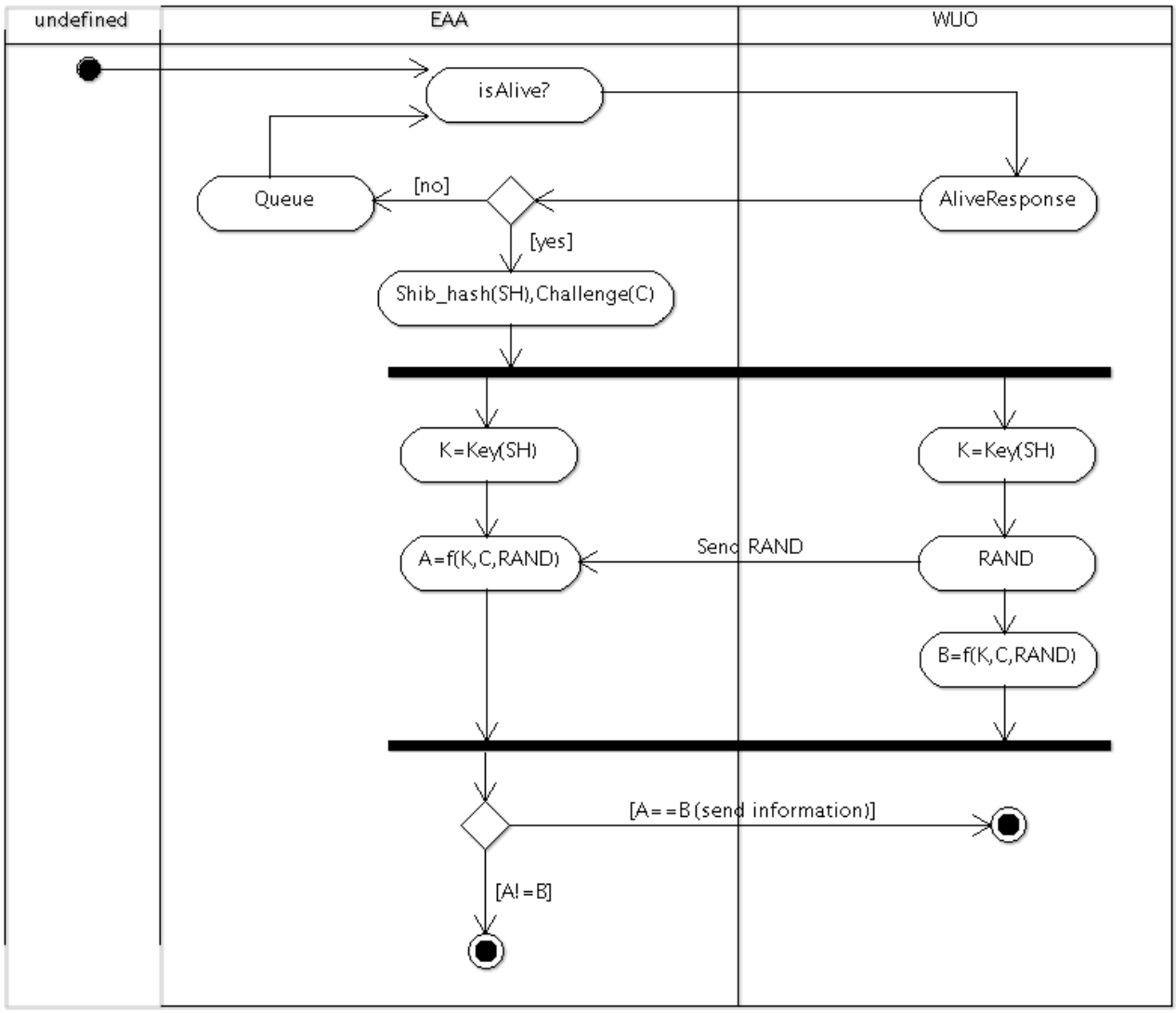
In order to ensure that a user exists at a WUO and at the same time to constrain the information distribution, cryptographic measures will be used to control the information flow. A symmetric challenge-response authentication concept will be applied.

Besides the Shibboleth hash, the basis for this mechanism is a key associated with the user. The key must always remain secret and protected. It is transported to the WUO by the user as he matches his EAA-account with his WUO-account.

To ensure that a user exists at a WUO, the users Shibboleth hash is sent together with a challenge to the WUO. The WUO retrieves the specific user key from its local database and generates a random number, which is sent back to the EAA. Now both EAA and WUO apply a function to the key, the challenge and the random number and the WUO sends the answer back to EAA. Now EAA verifies the equality of both answers and then sends the updated information to the WUO. An answer must be returned in a timeout period in order to prevent attacks.

Variable	Description
Shib_hash(SB)	Shibboleth hash used for exact user matching
Challenge(C)	Random element used as challenge
Key(K)	A key associated with the Shib_hash(SB)
f()	Cryptographic function, e.g. SHA-2
RAND	Random element used as client challenge
A	Umbrella generated string
B	WUO generated string

Following figure explains this workflow:



There are existing libraries available from:

https://github.com/Umbrella-Commiters/UmbrellaFrontend/tree/master/src/main/php

6.5.3 Umbrella Account Upgrade

The Umbrella Account Upgrade functionality is a shortcut from a WUO to the Umbrella system, which assists the users in creating an Umbrella account.

The idea is to have a button inside every WUO that when clicked on it transmits the information relevant to create an Umbrella account to the Umbrella system and displays it in its registration form.

The URL to call looks as follows:

```
https://umbrellaid.org/euu/account/create
```

It accepts following POST and GET attributes:

- username
- mail
- bdate

6.5.4 Graphical elements: conditions and status

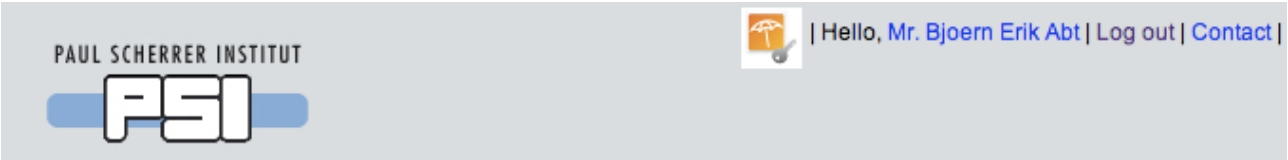
To further help users with the Umbrella system we propose following graphical helper elements depending on the condition the user is in:

Condition	Status
• User not logged in	• Login to WUO • Login to Umbrella
• Logged in at WUO • No account matching with Umbrella	• Upgrade to Umbrella
• Logged in at WUO • Accounts matched • Not logged in at Umbrella	• Login to Umbrella
• Logged in at Umbrella • Accounts matched • Logged in at WUO	• Logged in at Umbrella

These elements should be displayed close to the user name in the WUO. As an example we show an implementation done at DUO:

Login to Umbrella

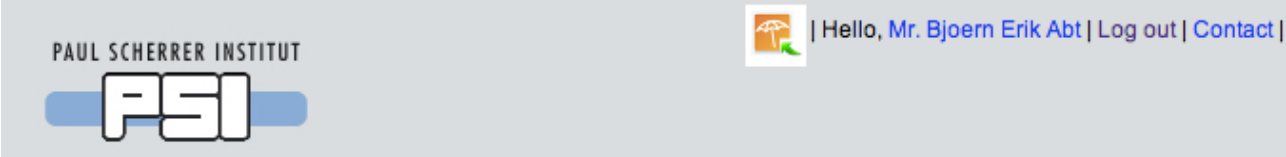
This image shows the icon to be displayed when the user is not logged in at all:



And the icon for itself:

Upgrade to Umbrella

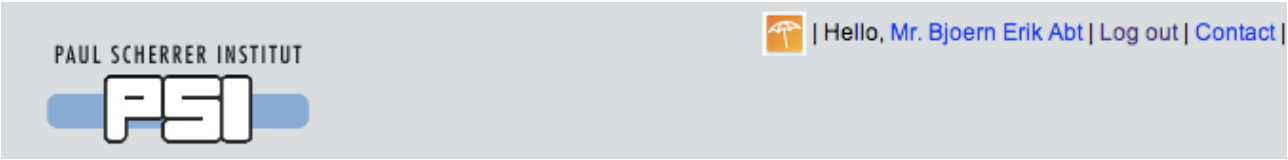
This should be shown when the user is logged in to the WUO but not with Umbrella:



And the icon for itself:

Logged in at Umbrella

This should be shown when the user is logged in at the WUO using Umbrella:



And the icon for itself:

6.6 Operations

6.6.1 Local backup

The local backup of the service provider is relatively simple. We just need to backup all the files located under `/etc/shibboleth`. There are no persistent user data to take care of.

Any adjustment done to the WUO or any other application must be backed-up separately as well as all additional user data stored in a database. Normally these applications already have an existing backup procedure and this could certainly be used.

6.6.2 Logfiles

All service provider logfiles are located under `/var/log/shibboleth`.

Following logfiles are written:

File	Description
shibd.log	The is the main log file that traces the software. Most of the real work is done by shibd and ends up in this file.
shibd_warn.log	The same as shibd.log except that it just contains logentries with a severity of WARN or higher.
transaction.log	The transaction log is a special log file created to track information that may be useful if you need to audit who is accessing you site or look up information to cross reference in the other logs. It tracks all sessions as they are created or deleted, and the attribute query operations that succeed or fail.

The log level can be changed in following files under `/etc/shibboleth`:

- shibd.logger
- syslog.logger
- native.logger
- console.logger

The files contain a log4j syntax.

6.6.3 Monitoring

Following URL can be used to monitor a Shibboleth service provider:

```
https://<YOUR_SP_URL>/Shibboleth.sso/Status
```

By default the URL is restricted to be only accessible by localhost. This can be extended by changing following in the file `/etc/shibboleth/shibboleth2.xml`:

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

to

```
<Handler type="Status" Location="/Status" acl="<YOUR_IP>"/>
```

As an addition you should do SNMP monitoring of free disk space, free RAM, etc. of the service provider server itself.

6.6.4 Upgrade procedure

The upgrade of the service provider can be done via the upgrade facility of the operating system:

```
# apt-get update
# apt-get upgrade
```

For major releases there can be changes in the configuration files which must be resolved by hand.

Bibliography

OASWS: OASIS, https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security, 2005,
UMBWS: van Daalen, Mirjam - Weyer, Heinz-Josef - Abt, Björn Erik, https://umbrellaid.org, 2013,
RFC4511: J. Sermersheim, https://tools.ietf.org/rfc/rfc4511.txt,
ODJWS: ForgeRock, http://forgerock.com/, 2014,
ODJDS: ForgeRock, http://forgerock.org/downloads/opendj-archive/, 2014,
RFC2605: G. Mansfield - S. Kille, http://tools.ietf.org/html/rfc2605,
NIIFSLO: -, https://wiki.aai.niif.hu/index.php/Single_Logout_in_Shibboleth_IdP,