Installation procedure for eduGAIN Access Check

Refer to the architecture document to have an overview of the service.

1 Prerequisites

- mysql-server
- httpd (Apache)
- mod_ssl

2 simpleSAMLphp IdP

- see simpleSAMLphp installation documentation

2.1 Prerequisites

- php
- php-xml
- php-mcrypt
- php-mysql

2.2 Installation

```
cd /opt/testidp
tar -zxvf src/simplesamlphp-1.13.1.tar.gz
ln -s simplesamlphp-1.13.1 simplesamlphp
```

/etc/httpd/conf.d/my-vhost.conf

```
Alias /simplesaml /opt/testidp/simplesamlphp/www
```

2.3 Configuration

2.3.1 enabling the IdP

/opt/testidp/simplesamlphp/config/config.php

```
    'auth.adminpassword'        => 'skjh8sJ/lkj',

    'secretsalt' => '*2osPG)RC?jf-t',

    'technicalcontact_name'     => 'IdP admin',
    'technicalcontact_email'    => 'idp-admin@my.fqdn',

    'enable.saml20-idp'        => true,
    'enable.shib13-idp'        => true,

    'debug' => TRUE,

    'logging.level'         => SimpleSAML_Logger::DEBUG,
```

2.3.2 enabling authentication modules

```
touch modules/exampleauth/enable
touch modules/authcrypt/enable
```

Later on we will configure loading of test accounts definition through a dynamically generated file.

2.3.3 configuring X509 certificates for SAML

```
mkdir cert
openssl req -newkey rsa:2048 -new -x509 -days 3652 -nodes -out
cert/idp-saml.crt -keyout cert/idp-saml.pem
```

metadata/saml20-idp-hosted.php

```
    /* X.509 key and certificate. Relative to the cert directory. */
    'privatekey' => 'idp-saml.pem',
    'certificate' => 'idp-saml.crt',
```

2.3.4 using OIDs as user attributes identifiers

metadata/saml20-idp-hosted.php

```
    /* Uncomment the following to use the uri NameFormat on attributes.
*/
    'attributes.NameFormat' =>
'urn:oasis:names:tc:SAML:2.0:attrname-format:uri',
    'authproc' => array(
        // Convert LDAP names to oids.
        100 => array('class' => 'core:AttributeMap', 'name2oid'),
    ),
```

2.3.5 generating eduPersonTargetedID

- simpleSAMLphp related documentation: https://simplesamlphp.org/docs/1.12/core:authproc_targetedid

We configure the IdP to release eduPersonTargetedID as a nameID, as expected by Shibboleth SPs.

metadata/saml20-idp-hosted.php

```
    'authproc' => array(
        60 => array(
                'class' => 'core:TargetedID',
                'nameId' => TRUE,
        ),

        // Convert LDAP names to oids.
        100 => array('class' => 'core:AttributeMap', 'name2oid'),
    ),

    'attributeencodings' => array(
                      'urn:oid:1.3.6.1.4.1.5923.1.1.1.10' => 'raw', /*
eduPersonTargetedID with oid NameFormat. */
    ),
```

2.3.6 attributes filtering

- core::attributeLimit documentation

Release attributes that the SP requested in its SAML metadata via RequestedAttribute XML entries:

config/config.php

```
        // We need this attribute mapping before AttributeLimit
        // otherwise SPs attributes expectations listed in MD can't be
used
            49 => array('class' => 'core:AttributeMap', 'name2oid'),


         50 => array(
                /* Default: release no attribute
               unless SP metadata mentions requested attributes */
                'class' => 'core:AttributeLimit',
                'default' => TRUE,
         ),
```

2.3.7 configuring logs

simplSAMLphp uses local5 as the default log facility.

/etc/syslog-ng/syslog-ng.conf

```
## simpleSAMLphp
destination simplesamlphp { file ("/var/log/simplesamlphp") ; };
filter f_filter f_simplesamlphp     { facility(local5)  and
match('simple'); };
log { source(src); filter(f_simplesamlphp); destination(simplesamlphp);
};
```

2.3.8 loading federation metadata

In the following examples we load eduGAIN metadata, directly from the MDS. Adapt metadata configuration to your use case.

```
touch modules/cron/enable
cp modules/cron/config-templates/*.php config/
touch modules/metarefresh/enable
cp modules/metarefresh/config-templates/*.php config/
```

config/module_cron.php

```
$config = array (
    'key' => 'zNddh3V5',
    'allowed_tags' => array('daily', 'hourly', 'frequent'),
    'debug_message' => TRUE,
    'sendemail' => FALSE,

);
```

Go to https://my.fqdn/simplesaml/module.php/cron/croninfo.php to get the command-line calls you should install in the crontab. You will be asked for the admin password you've defined earlier in config.php file.

If you paste these to /etc/crontab, you should add the user name (root) to each command line, as shown is the example below:

/etc/crontab

```
## simpleSAMLphp : loading of federation metadata
02 0 * * * root curl --insecure --silent
"https://my.fqdn/simplesaml/module.php/cron/cron.php?key=zNddh3V5&tag=daily" > /dev/null 2>&1
01 * * * * root curl --insecure --silent
"https://my.fqdn/simplesaml/module.php/cron/cron.php?key=zNddh3V5&tag=hourly" > /dev/null 2>&1
14 2 * * * root curl --insecure --silent
"https://my.fqdn/simplesaml/module.php/cron/cron.php?key=zNddh3V5&tag=frequent" > /dev/null 2>&1
```

The curl --insecure options disables server certification check

| config/config-metarefresh.php

```
    'sets' => array(

        'edugain' => array(
            'cron'        => array('hourly'),
            'sources'     => array(
                array(
                    /*
                     * entityIDs that should be excluded from this src.
                     */
                    #'blacklist' => array(
                #    'http://some.other.uni/idp',
                #),

                    /*
                     * Whitelist: only keep these EntityIDs.
```

```
                         */
                        #'whitelist' => array(
                    #    'http://some.uni/idp',
                    #    'http://some.other.uni/idp',
                    #),

                        #'conditionalGET' => TRUE,
                    'src' => 'http://mds.edugain.org/',
                        'validateFingerprint' =>
'3A:84:61:CE:E5:91:99:B4:07:73:9A:83:3A:20:59:59:43:1B:FE:FF',
                        'template' => array(
                            'tags'    => array('edugain'),
                            'authproc' => array(
                                51 => array('class' => 'core:AttributeMap',
'oid2name'),
                            ),
                        ),
                    ),
                ),
                'expireAfter'         => 60*60*24*4, // Maximum 4 days
cache time.
                'outputDir'     => 'metadata/metadata-edugain/',

                /*
                 * Which output format the metadata should be saved as.
                 * Can be 'flatfile' or 'serialize'. 'flatfile' is the
default.
                 */
                'outputFormat' => 'flatfile',
            ),
        ),
```

to calculate the fingerprint of a certificate, check https://simplesamlphp.org/docs/1.5/simplesamlphp-reference-idp-remote#section_5

```
mkdir metadata/metadata-edugain
chown apache metadata/metadata-edugain
```

| config/config.php

```
    'metadata.sources' => array(
        array('type' => 'flatfile'),
        array('type' => 'flatfile', 'directory' =>
'metadata/metadata-edugain'),
    ),
```

2.4 Registering the IdP in the federation

You should register the IdP in your local federation, or in eduGAIN.

Check the IdP metadata to get the SAML endpoints:
https://my.fqdn/simplesaml/saml2/idp/metadata.php

2.5 Enabling user consent

- simpleSAMLphp documentation: https://simplesamlphp.org/docs/1.13/consent:consent

Enable the module:

```
touch modules/consent/enable
```

Define the userid attribute:

metadata/saml20-idp-hosted.php

```
    /* UserID used by the consent module */
    'userid.attribute' => 'uid',
```

Enabling the consent filter:

config/config.php

```
        /*
         * Consent module is enabled (with no permanent storage, using
cookies).
      */

        90 => array(
            'class' => 'consent:Consent',
            'store' => 'consent:Cookie',
            'focus' => 'yes',
            'checked' => TRUE
        ),
```

2.6 Advanced configuration

- see simpleSAMLphp authentication processing filters documentation

Each test account is bound to a SP and should be used to login at this SP only. That behavior is the main specificity to the eduGAIN Access Check IdP compared to a standard IdP.

This behavior has been added through an authentication processing filter using the **restrictLogin** simpleSAMLphp module developped by RENATER. The **restrictLogin** simpleSAMLphp is not distributed with simpleSAMLphp.

Installing and enabling the module:

```
svn export
svn+ssh://svn@svn.geant.net/GEANT/edugain_testidp_account_manager/trunk/othe
r/restrictLogin_SSP_module.tar.gz /tmp/restrictLogin_SSP_module.tar.gz
cd simplesaml/modules
tar -zxvf /tmp/restrictLogin_SSP_module.tar.gz
touch restrictLogin/enable
```

Configure use of the authentication processing filter:

config/config.php

```
    47 => array(
        'class' => 'restrictLogin:RestrictLogin',
        'sp_entityid_attr' => 'associatedSP',
    ),
```

The processing filter makes use of a special user attributes: associatedSP. This user attributes is provisionned by the account manager and refers to the entityID of the SP that is associated to each test user account.

2.7 Enable MetaMerge to filter user attributes based on entity categories

It is expected that eduGAIN Access Check has different attribute release rules for SPs that belong to entity categories (http://refeds.org/category/research-and-scholarship or http://www.geant.net/uri/dataprotection-code-of-conduct/v1).

Proposed attribute release policy

The proposed attribute release policy can be summarized as follows:

1. SP supports CoCo
    1. release only required RequestedAttributes amongst
        1. cn
        2. displayName
        3. eduPersonAffiliation

        4. eduPersonScopedAffiliation
        5. mail
        6. schacHomeOrganization
        7. schacHomeOrganizationType
        8. eduPersonPrincipalName
        9. eduPersonTargetedID
2. SP supports R&S
    1. release only required+optional RequestedAttributes amongst
        1. cn
        2. displayName
        3. eduPersonAffiliation
        4. eduPersonScopedAffiliation
        5. mail
        6. eduPersonPrincipalName
        7. eduPersonTargetedID
3. SP supports none
    1. release only mandatory RequestedAttributes amongst
        1. eduPersonAffiliation
        2. eduPersonScopedAffiliation
        3. schacHomeOrganization
        4. schacHomeOrganizationType
        5. eduPersonTargetedID

Implementing the attribute release policy

simpleSAMLphp 1.13.2 does not provide configuration options to define such an attribute policy. Options to do it:

1. https://github.com/simplesamlphp/simplesamlphp/issues/49: a patched version of AttributeLimit filter (November 2013), submitted by Brook Schofield
    - the simpleSAMLphp developpers suggested that Brook turned the code into a new module.
        - we need to wait for Brook to rewrite the code
2. https://github.com/gollmann/MetaMerge: a metamerge module by G.Gollmann (September 2014).
    - metamerge module transforms simpleSAMLphp's saml20-sp-remote.php to reflect the attribute release policy. The policy is defined in a saml20-sp-mixin.php. It allows to define different set of attributes to be released, depending on SP entityIDs, SP entity categories, attributes requirements (versus optional attributes).
    - a git pull request https://github.com/simplesamlphp/simplesamlphp/pull/105 has been submitted to simpleSAMLphp developpers to keep track of required attributes in saml20-sp-remote.php.
        - this patch should be applied to our simpleSAMLphp 1.13.2

Installing MetaMerge module

We gave up the idea of using this module. It requires patching of both simpleSAMLphp 1.13 (to get the attributes.requred info populated) and the module itself (mergeMetadata.php)

Download MetaMerge module:

```
cd /opt/testidp/src
git clone https://github.com/gollmann/MetaMerge.git MetaMerge
```

Install and enable the module:

```
cp -pR /opt/testidp/src/MetaMerge/metamerge
/opt/testidp/simplesamlphp/modules/
touch /opt/testidp/simplesamlphp/modules/metamerge/enable
```

If you run php < 5.4, apply the following patch:

```
*** modules/metamerge/bin/mergeMetadata.php 2014-12-11 10:49:33.000000000
+0100
--- /opt/testidp/src/MetaMerge/metamerge/bin/mergeMetadata.php  2014-12-08
14:31:09.000000000 +0100
***************
*** 36,42 ****

  //    map "attributes" and "attributes.required" from OIDs to friendly
names
    $mapper = new sspmod_core_Auth_Process_AttributeMap(array('oid2name'),
NULL);
!   foreach(array('attributes', 'attributes.required') as $field) {
        if(isset($mdata[$field])) {
            $tmp = array('Attributes' => array_fill_keys($mdata[$field],
0));
            $mapper->process($tmp);
--- 36,42 ----

  //    map "attributes" and "attributes.required" from OIDs to friendly
names
    $mapper = new sspmod_core_Auth_Process_AttributeMap(array('oid2name'),
NULL);
!   foreach(['attributes', 'attributes.required'] as $field) {
        if(isset($mdata[$field])) {
            $tmp = array('Attributes' => array_fill_keys($mdata[$field],
0));
            $mapper->process($tmp);
```

Other changesare also needed in the code
To do... include patches

Configure source for metadata and destination directory:

modules/metamerge/config/config.php

```
<?php
```

```
$fingerprint =
'3A:84:61:CE:E5:91:99:B4:07:73:9A:83:3A:20:59:59:43:1B:FE:FF';
$metadatasource = 'http://mds.edugain.org/';
$destination = 'metadata-edugain/saml20-sp-remote.php'

?>
```

Define the transformation process:

modules/metamerge/config/saml20-sp-mixin.php

```
<?php

$fieldsToStrip = array('entityDescriptor');

// NB: The examples assume that the authsource delivers attributes
using friendly names
// and that the following global attribute filters are configured:
//        50 => 'core:AttributeLimit',
//        90 => array( 'class' => 'consent:Consent', ...),
//        95 => array('class' => 'core:AttributeMap', 'name2oid'),

$defaultTemplate = array (
   'attributes.allowed.ifRequired' => array('eduPersonAffiliation',
'eduPersonScopedAffiliation', 'schacHomeOrganization',
'schacHomeOrganizationType', 'eduPersonTargetedID'),
);

// Service Provider
/*
$template['https://skriptenforum.net/shibboleth'] = array (
  'attributes' => array('eduPersonScopedAffiliation'),
);
*/

// Service Categories

$template['http://refeds.org/category/research-and-scholarship'] =
array (
  'attributes.allowed' => array('cn','displayName',
'eduPersonAffiliation','eduPersonScopedAffiliation', 'mail',
'eduPersonPrincipalName','eduPersonTargetedID'),
);

$template['http://www.geant.net/uri/dataprotection-code-of-conduct/v1']
= array (
   'attributes.allowed.ifRequired' => array('cn','displayName',
'eduPersonAffiliation','eduPersonScopedAffiliation', 'mail',
'schacHomeOrganization',  'schacHomeOrganizationType',
'eduPersonPrincipalName','eduPersonTargetedID'),
);

?>
```

Run the merge process via crontab:

```
php modules/metamerge/bin/fetchFederationMetadata.php
```

Script should be run as 'apache' user, otherwise saml20-sp-remote.php belongs to root user.

3 Test account manager

The test accounts manager is the application responsible for creating, removing test accounts. It has been developped by RENATER.

The code is public; you can access the repository: svn+ssh://svn@svn.geant.net/GEANT/edugain_testidp_account_manager

The application is written in Perl using the following specific CPAN modules:

- Template: template toolkit is used to write templates (mail, web, test account).
  - Check the Template toolkit documentation.
- Rose::DB::Object: a Perl ORM (Object Relationnal Mapper) used to automatically generate the RDBMS code.
  - Check Rose::DB::Object documentation
  - RDBMS supported by the library are: Pg, mysql, SQLite, Informix, Oracle
- XML::LibXML: an XML parser used to parse SAML metadata

The web GUI of the application uses the following frameworks:

- JQuery
- JQuery UI
  - http://jqueryui.com/autocomplete/ autocomplete widget provides autocompletion in the SP selection menu
- JQuery-Steps: a wizard plugin
- JQuery validation: a form validation plugin

3.1 Prerequisites

The following CPAN Perl modules should be installed:

- perl-CPAN

- perl-Template-Toolkit
- perl-Unicode-MapUTF8
- perl-MIME-EncWords
- perl-MIME-Charset
- perl-XML-LibXML
- perl-Rose-DB-Object

3.2 Installing the code

```
cd /opt/testidp
svn co
svn+ssh://svn@svn.geant.net/GEANT/edugain_testidp_account_manager/trunk
IdPAccountManager
chown -R apache IdPAccountManager
```

Apache user needs:

- read access to all files,
- write access to conf/ and log/

Copy the sample Conf.pm configuration file and customize it:

```
cd IdPAccountManager
cp conf/default-Conf.pm conf/Conf.pm
```

conf/Conf.pm

```
our %global = (

    ## Code version
    'version' => 'closed Beta 1',

    ## Name of the application used in web pages, mail notices
    'app_name' => 'eduGAIN Access Check',

    ## URL of the application
    'app_url' => 'https://my.fqdn/accountmanager',

    ## Validity period of test accounts, in days
    'accounts_validity_period' => 7,

    ## Scope used by the Test IdP
    'idp_scope' => 'my.fqdn',

    ## EntityID of the IdP
    'idp_entityid' =>
'https://my.fqdn/simplesaml/saml2/idp/metadata.php',

    ## Name of the IdP
    'idp_displayname' => 'eduGAIN Access Check',

    ## Root simpleSamlPhp directory
    'root_ssp_dir' => '/opt/testidp/simplesamlphp',

    ## Root test account manager directory
    'root_manager_dir' => '/opt/testidp/IdPAccountManager',

    ## Database type refers to a Perl Database Driver name
    ## However only a subset of existing DBDs are supported by
Rose::DB::Object:
    ## Pg, mysql, SQLite, Informix, Oracle (DBD names are case
sensitives)
    'database_type' => 'mysql',

    ## Database hostname
    'database_host' => 'localhost',

    ## Database_name
    'database_name' => 'idp_account_manager',

    ## Database username
    'database_user' => 'idpadmin',

    ## Database user password
    'database_password' => 'secret',

    ## Log file for the manager
    'log_file' => '/opt/testidp/IdPAccountManager/log/manager.log',

    ## Log level : debug, info, trace, notice, error
    'log-level' => 'info',

    ## email address to contact admins
    'admin_email' => 'john@my.fqdn',

    ## email address to ask for support
    'support_email' => 'support@my.fqdn',

    ## Development feature
    ## Protection to prevent notifications during test dev phases
    ## Notify only admin_email above
    'dev_no_mail_outside' => 1,

    ## Development feature
    ## hard-coded list of contactPersons
    ## these email addresses will be added to the list of contacts for
any SP
    'dev_sp_contact' => 'john@my.fqdn,sarah@my.fqdn',

    ## From field use by the account manager
    'notice_from' => 'testidpaccountmanager@my.fqdn',

    ## federation metadata local copy path
    'federation_metadata_file_path' =>
'/opt/testidp/IdPAccountManager/conf/edugain-md.xml',

    ## valid account profiles
    'account_profiles' => ['student1','teacher1'],
);
```

3.3 Software organization

The software is organized as follows:

- bin/: binaries
  - account-manager-client.pl: a command-line client
  - account-manager-web.pl: the web interface CGI script
- conf/: configuration files
  - Conf.pm: the main configuration file
  - create-manager-db.sql: the SQL DB creation script
- doc/: documentation
- lib: libraries
- log: log directory
- resources: static web content (icons, CSS, JS libraries)
- templates:
  - accountProfiles: test account profile templates
  - mail: mail notice templates
  - web: web pages templates

3.4 Apache setup

/etc/httpd/conf.d/my-vhost.conf

```
ScriptAlias /accountmanager
/opt/testidp/IdPAccountManager/bin/account-manager-web.pl
Alias /resources /opt/testidp/IdPAccountManager/resources
```

3.5 Database setup

The test account manager stores data in a relational database. You should create the database; here is an example for MySQL:

```
# mysqladmin create idp_account_manager
# mysql idp_account_manager < conf/create-manager-db.sql
```

The low-level code to interface with the RDBMS is handled by an ORM (Object Relational Mapping) Perl library (Ose::DB::Object). However only a subset of existing Perl Database Drivers are supported by Rose::DB::Object (Pg, mysql, SQLite, Informix, Oracle). The RDBMS code needs to be updated to interface with your local DB.

Customize the conf/Conf.pm file to provide your RDBMS details:

```
    ## Database type refers to a Perl Database Driver name
    ## However only a subset of existing DBDs are supported by
Rose::DB::Object:
    ## Pg, mysql, SQLite, Informix, Oracle (DBD names are case sensitives)
    'database_type' => 'mysql',

    ## Database hostname
    'database_host' => 'localhost',

    ## Database_name
    'database_name' => 'idp_account_manager',

    ## Database username
    'database_user' => 'idpadmin',

    ## Database user password
    'database_password' => 'xxxx',
```

bin/create-database.pl script will analyse your database structure to generate appropriate Perl code for DB access.
Run bin/create-database.pl script and install the generated code in the lib/ directory:

```
# ./bin/create-database-code.pl
Database-related code created in /tmp/IdPAccountManager. You should copy
this code in lib/ directory

# cp -R /tmp/IdPAccountManager/Data lib/IdPAccountManager/
```

3.6 Loading federation metadata

Add this curl call to the crontab to download the federation metadata twice a day:

```
# eduGAIN test account manager metadata download
18 8,13 * * * root curl http://mds.edugain.org/ -z
/opt/testidp/IdPAccountManager/conf/edugain-md.xml --output
/opt/testidp/IdPAccountManager/conf/edugain-md.xml --silent || echo "Failed
to download eduGAIN metadata"
```

3.7 Accessing the account manager

You can now try accessing https://my.fqdn/accountmanager

3.8 Configuring simpleSAMLphp to use test accounts

simpleSAMLphp will loads its user accounts through the inclusion a flat file (conf/valid-accounts.php) in config/authsources.php. You should edit config/authsources.php to remove the previously configured “student” and “employee” accounts and include the conf/valid-accounts.php file. Here is what config/authsources.php should look like:

config/authsources.php

```php
<?php

include "/opt/testidp/IdPAccountManager/conf/valid-accounts.php";

$config = array(

...

    'example-userpass' => $validTestAccounts,
```

The included valid-accounts.php file is automatically generated by the Test account manager, every time test accounts are created/removed.

Fo each test account, valid-accounts.php file provides the user ID, password hash and a set of user attributes, derived from a test account profile. Here is a sample valid-accounts.php file:

valid-accounts.php

```php
<?php
// template for a PhP configuration file loaded in simpleSamlPhp
authsources.php file
$validTestAccounts = array (
    'authcrypt:Hash',

  'user319:{SHA256}YvNJqmd3VFjqvqDYcyTcXcr8zZdhkuhALTbyMrgJtF4=' =>
array(
        'cn' => 'John Kleinman',
        'displayName' => 'John Kleinman',
        'eduPersonAffiliation' => array('member','student'),
        'eduPersonPrincipalName' => '319@dev-edugain.renater.fr',
        'eduPersonScopedAffiliation' => array(
'member@dev-edugain.renater.fr','student@dev-edugain.renater.fr'),
        'eduPersonTargetedID' =>
'https://dev-edugain.renater.fr/simplesaml/saml2/idp/metadata.php!https:
//test.federation.renater.fr/test/ressource!X622UR2A7PG1uVhATobBOrMz+Ys
=',
        'mail' => 'john.kleinman@dev-edugain.renater.fr',
        'schacHomeOrganization' => 'dev-edugain.renater.fr',
        'schacHomeOrganizationType' =>
'urn:schac:homeOrganizationType:int:university',
        'uid' => '319',

        'associatedSP' =>
"https://test.federation.renater.fr/test/ressource",
  ),

  'user320:{SHA256}3cb3hdyg2VYHiVqlaWXVukp95VG8OMjYPp2SUKUFH8g=' =>
array(
        'cn' => 'Peter Smith',
        'displayName' => 'Peter Smith',
        'eduPersonAffiliation' => array('member','faculty'),
        'eduPersonPrincipalName' => '320@dev-edugain.renater.fr',
        'eduPersonScopedAffiliation' => array(
'member@dev-edugain.renater.fr','student@dev-edugain.renater.fr'),
        'eduPersonTargetedID' =>
'https://dev-edugain.renater.fr/simplesaml/saml2/idp/metadata.php!https:
//test.federation.renater.fr/test/ressource!X622UR2A7PG1uVhATobBOrMz+Ys
=',
        'mail' => 'peter.smith@dev-edugain.renater.fr',
        'schacHomeOrganization' => 'dev-edugain.renater.fr',
        'schacHomeOrganizationType' =>
'urn:schac:homeOrganizationType:int:university',
        'uid' => '320',

        'associatedSP' =>
"https://test.federation.renater.fr/test/ressource",
  ),
```

3.9 Testing the Test IdP

Now that test accounts created by the manager are loaded by simpleSAMLphp, you can try creating accounts and using them to login at your SP: https://my.fqdn/accountmanager

3.10 Programming expiration of test accounts and validation tokens

Test accounts get automatically expired after N days; the validity period is defined in conf/Conf.pm:

conf/Conf.pm

```
    ## Validity period of test accounts, in days
    'accounts_validity_period' => 7,
```

Authentication tokens that have not been confirmed get automatically expired after N hours; the validity period is defined in conf/Conf.pm:

conf/Conf.pm

```
    ## Token validity period, in hours
    'tokens_validity_period' => 2,
```

You should add the following entry to your crontab:

/etc/crontab

```
## Test IdP - removed expired tokens and accounts
46 7 * * * root
/opt/testidp/IdPAccountManager/bin/account-manager-client.pl
--list_test_accounts --filter_expired --delete > /dev/null
29 * * * * root
/opt/testidp/IdPAccountManager/bin/account-manager-client.pl
--list_authentication_tokens --filter_expired --delete > /dev/null
```