Proxy for Vidyo

This wiki page explains how SimpleSAML PHP can be used to enable federated login for the VidyoPortal video conferencing system. Since version 3.0, Vidyo has enabled support for SAML in the VidyoPortal. They do however not rely on existing SAML middleware (such as Shibboleth or SimpleSAMLphp) but they manage SAML directly, which has some drawbackes when it comes to IdP discovery and metadata handling in case users of hundreds of IdPs should be allowed to log in via SAML.

At the time of writing, SAML is supported for a single tenant, ie. one single Identity Provider can be configured in VidyoPortal. This means that in order to make Vidyo as a federated service, a SAML gateway should be configured to provide access for institutional users. In this guide, SimpleSAMLphp will be used as a gateway.

Even though this guide focuses on how to enable SAML login from many IdPs for Vidyo, it can also serve as an example for other applications that "support SAML" but only on a small scale with one IdP at once.

Install SimpleSAMLphp

To deploy the SAML proxy, we recommend to use the popular SimpleSAMLphp SAML implementation, which is quite versatile and is frequently used in academic identity federations. The installation of SimpleSAMLphp software is covered in its documentation.

You will need to configure SSP both as an IdP (in the direction of VidyoPortal) and an SP (for the federation). Contrary to what is stated in section 9.1 of the SimpleSAMLphp IdP documentation, both IdP and SP should share the same hostname / virtual host.

Configure the IdP part

Your proxy will be an IdP from the point of view of the VidyoPortal. The IdP will use the SP part for authentication, thus you can login to the proxy by the federation.

Enable SAML2 IdP functionality in `config/config.php`:

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

Generate a key and a long-living self-signed certificate for the IdP and place it in the `cert` directory as `idp.key` and `idp.crt`

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

Edit `metadata/saml20-idp-hosted.php` as the following:

```
'auth' => 'default-sp',
'privatekey' => 'idp.key',
'certificate' => 'idp.crt',
'authproc' => array(
   200 => array('class' => 'core:AttributeMap', 'oid2name'),
),
```

It is recommended to configure VidyoPortal by using friendly attribute names (you don't want to write OIDs), therefore use the built-in `oid2name`*AttributeMa* for the transformation of attribute names.

Retrieve VidyoPortal metadata from the portal administration interface and save it as `metadata/vidyo-sp.xml`. It needs to be referenced from `config/config.php`:

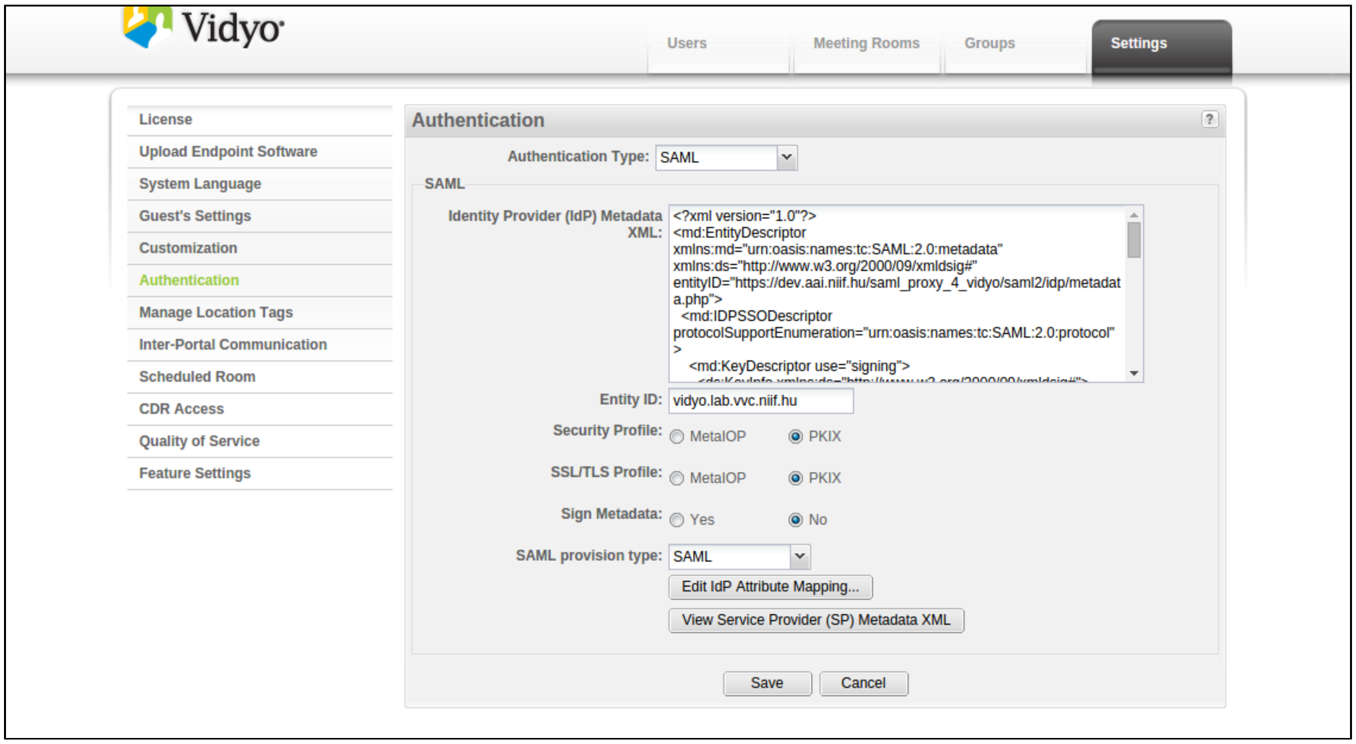
```
'metadata.sources' => array(
    ...
    array('type' => 'xml', 'file' => 'metadata/vidyo-sp.xml'), // vidyo sp
    ...
 ),
```

Configure VidyoPortal to use SAML

In the portal the following should be set:

- Authentication Type: **SAML**
- IdP Metadata XML: you can retrieve it from https://path.to.simplesaml.tld/saml2/idp/metadata.php

- SAML Provision type: **SAML** (automatic provisioning)



Authentication

Authentication Type: SAML

SAML

Identity Provider (IdP) Metadata XML:

```
<?xml version="1.0"?>
<md:EntityDescriptor
xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
entityID="https://dev.aai.niif.hu/saml_proxy_4_vidyo/saml2/idp/metadat
a.php">
  <md:IDPSSODescriptor
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol"
>
    <md:KeyDescriptor use="signing">
```

Entity ID: vidyo.lab.vvc.niif.hu

Security Profile: MetaIOP [✓ PKIX]

SSL/TLS Profile: MetaIOP [✓ PKIX]

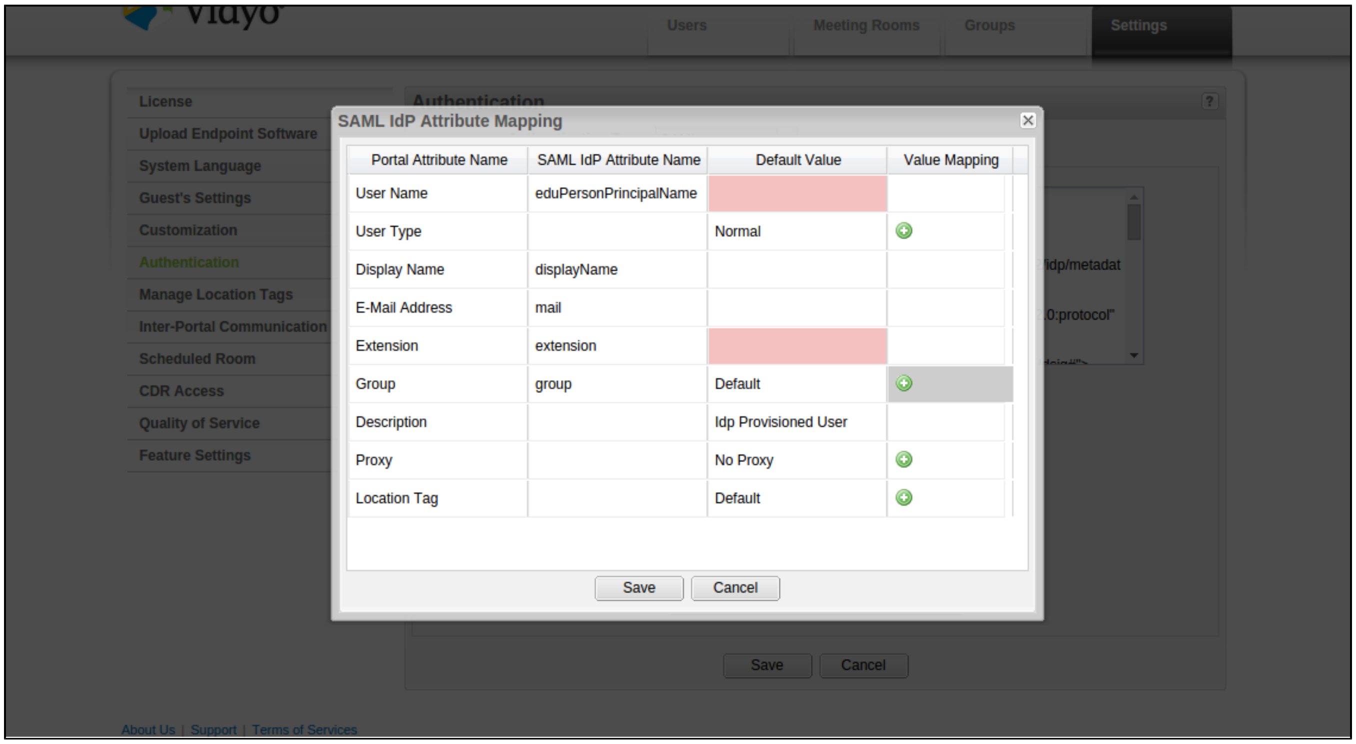
Sign Metadata: Yes [✓ No]

SAML provision type: SAML

Set Vidyo user parameters

Click on *Edit IdP Attribute Mapping...* for mapping the IdP attributes to Vidyo user parameters

The interface is pretty self-explanatory. You can edit which SAML IdP attributes can be mapped to certain Vidyo parameters.

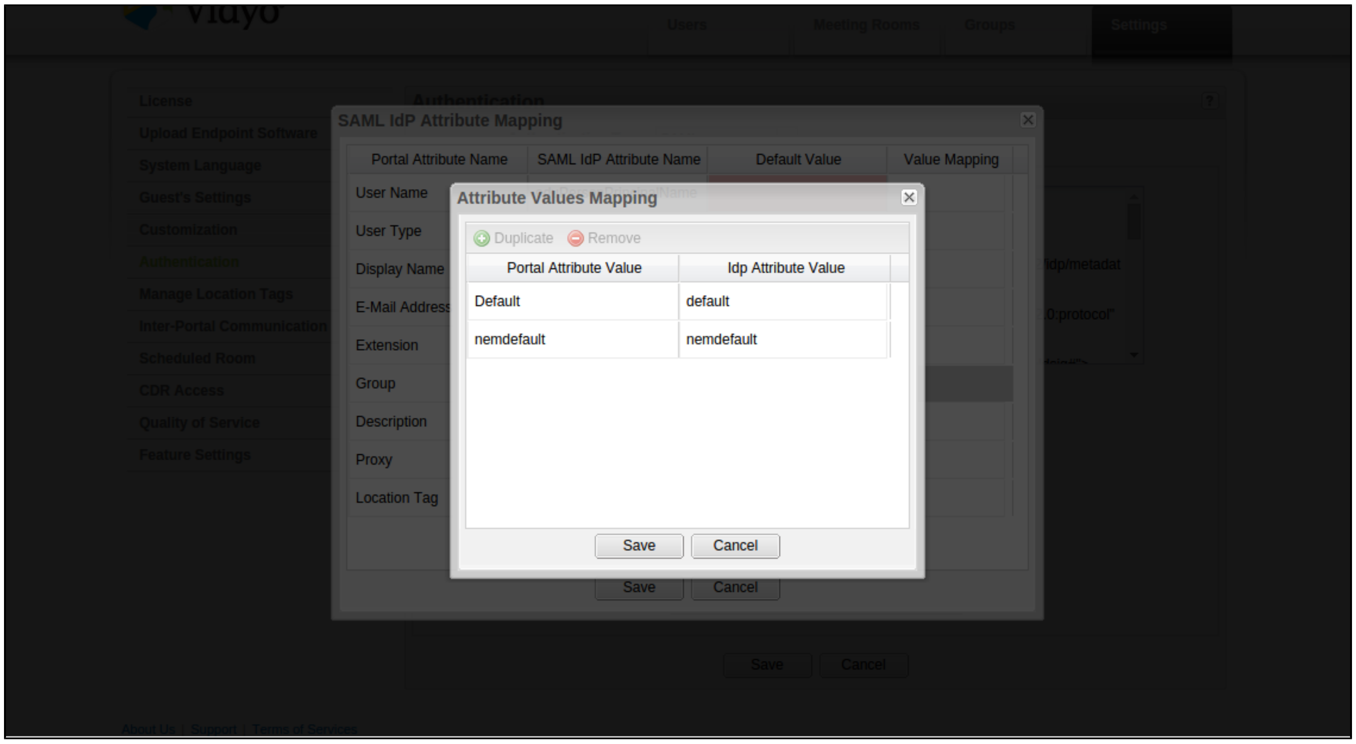


SAML IdP Attribute Mapping

Portal Attribute Name	SAML IdP Attribute Name	Default Value	Value Mapping
User Name	eduPersonPrincipalName	🟥	
User Type		Normal	⊕
Display Name	displayName		
E-Mail Address	mail		
Extension	extension	🟥	
Group	group	Default	⊕
Description		Idp Provisioned User	
Proxy		No Proxy	⊕
Location Tag		Default	⊕

Be aware that some fields have arbitrarily limited lengths, e.g. the user name is limited by Vidyo to 40 characters. If you map an attribute there whose values for a particular user are longer, than the user will receive misleading error messages ("No AuthenticationProvider found for org.springframework.security.saml.SAMLAuthenticationToken") after successful authentication to your SimpleSAMLphp proxy.

For some attributes it is possible to define value mapping as well. It might be useful for group and type information. You can define static string matches here.



SAML IdP Attribute Mapping

Portal Attribute Name	SAML IdP Attribute Name	Default Value	Value Mapping
User Name			
User Type			
Display Name			
E-Mail Address			
Extension			
Group			
Description			
Proxy			
Location Tag			

Attribute Values Mapping

Portal Attribute Value	Idp Attribute Value
Default	default
nemdefault	nemdefault

Configure the SP part

In the IdP configuration, you have referenced an authsource called *default-sp*. If you configure a SimpleSAMLphp SP with this identifier, the IdP settings above will direct your users to their home IdP.

For SP configuration, please follow your federation's guides or use SimpleSAMLphp documentation as a reference. Basically you will need to perform the following:

- generate a key and a certificate for the proxy (SP) and configure SSP to use them;
- register the proxy in your federation:
  - you will need to use the SP certificate here;
  - your attribute requirements depend on what attributes you have configured in Vidyo previously;
- configure the SP to refresh federation (or eduGAIN) metadata regularly by using *metarefresh*

Service specific terms of use (ToU)

If you wish your users to accept your service specific (or general) terms of use before using this service, you can implement that on the SimpleSAMLphp proxy after authentication on the user's home IdP. One easy way to achieve this is to configure and modify the SSP's consent module (you don't need it for attribute release anyway as this would be handled by the home IdP beforehand).

First, you need to enable the consent module by creating an empty file in the module directory:

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

We store the consent information in a database, so that the same user can come with different browsers / machines and his consent will still be recognized if he gave it once. To do that, you need to include the following in your authproc.idp array:

```
90 => array(
            'class' => 'consent:Consent',
            'store' => array(
               'consent:Database',
                'dsn' => 'mysql:host=localhost;dbname=consent',
                'username' => 'dbuser',
                'password' => 'dbuserpassword',
            ),
        ),
```

dbname, *dbuser* and *dbuserpassword* needs to be adapted to your environment.

In a multi-language environment, the LanguageAdaptor module should be configured in order to display the terms-of-use in the correct language for each user and to store the language information:

```
99 => 'core:LanguageAdaptor',
```

The next step is to disable the display of the attribute table on the consent dialogue. For this, you can just comment out lines 97 and 99 of `modules /consent/www/getconsent.php`, so that the section looks like this:

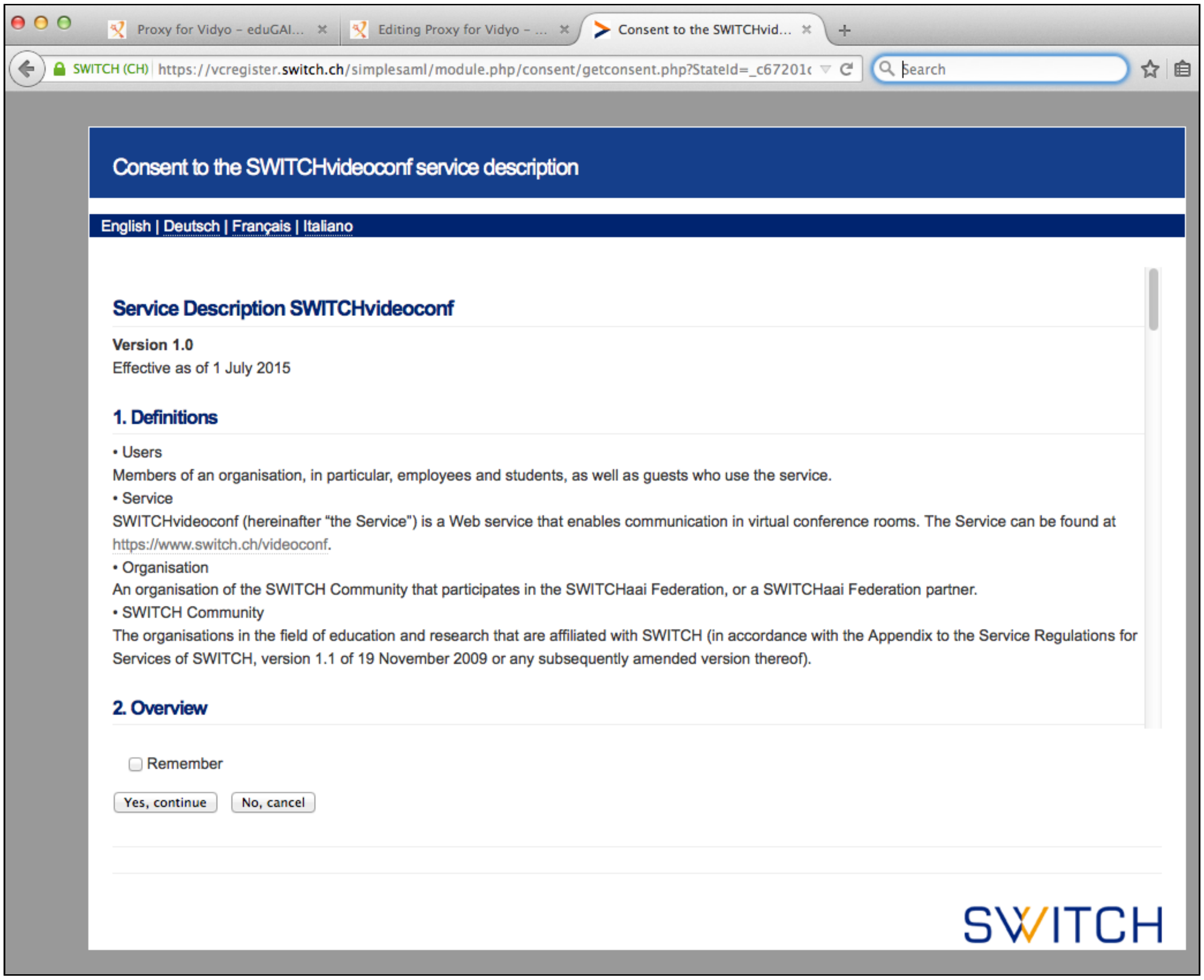
```
// Remove attributes that do not require consent: modified to remove all attributes
foreach ($attributes AS $attrkey => $attrval) {
//    if (in_array($attrkey, $noconsentattributes)) {
        unset($attributes[$attrkey]);
//    }
}
```

In order to change the displayed text, you need to go to the dictionaries folder of the consent module. `consent.definition.json` contains the English texts, while `consent.translation.json` contains the texts in all other necessary languages. You insert your terms of use inside the *consent_accept* fiel d. The file accepts html-tags, but everything has to be on one line without special characters (Umlauts and accented characters). We use BBEdits "Translate Text to HTML" feature to facilitate the creation of the string. In order to make it scrollable and keep the accept button visible at all times, you can wrap your text in a overflow-y div element. You should also adapt the *consent_header*, *noconsent_text*elements in both files for every language you want to use.

The last step is to limit the languages proposed by SimpleSAMLphp to the relevant ones for your environment. In Switzerland the section of the `config. php` looks like the following:

```
  /*
    * Languages available, RTL languages, and what language is default
    */
    'language.available' => array(
        'en', 'de', 'fr', 'it'
    ),
    'language.rtl' => array(),
    'language.default' => 'en',
```

The following image gives an example of the final result (after theming SSP for SWITCH).


Proxy for Vidyo – eduGAI...
Editing Proxy for Vidyo – ...
Consent to the SWITCHvid...
SWITCH (CH) https://vcregister.switch.ch/simplesaml/module.php/consent/getconsent.php?StateId=_c67201c
Search

Consent to the SWITCHvideoconf service description

English | Deutsch | Français | Italiano

Service Description SWITCHvideoconf

Version 1.0
Effective as of 1 July 2015

1. Definitions

• Users
Members of an organisation, in particular, employees and students, as well as guests who use the service.
• Service
SWITCHvideoconf (hereinafter "the Service") is a Web service that enables communication in virtual conference rooms. The Service can be found at https://www.switch.ch/videoconf.
• Organisation
An organisation of the SWITCH Community that participates in the SWITCHaai Federation, or a SWITCHaai Federation partner.
• SWITCH Community
The organisations in the field of education and research that are affiliated with SWITCH (in accordance with the Appendix to the Service Regulations for Services of SWITCH, version 1.1 of 19 November 2009 or any subsequently amended version thereof).

2. Overview

SimpleSAMLphp themes

In order to have SimpleSAMLphp come in your local look and feel, you can create a theme and exchange logos, favicons, colors, etc. This is documented here.

Authorization

Vidyo does not support fine-grained authorization of authenticated users. You can use the Authorize module of SimpleSAMLphp. The documentation can be found here.

Workarounds

The authors of this text used the following workarounds to make Vidyo as a production federated service in 2016q1. It is possible that future versions will make the workarounds unnecessary.

Flat attributes

Vidyo cannot handle multi-valued attributes. It just takes the first value and ignores the rest. For statistical purposes, SWITCH uses the eduPersonScopedAffiliation mapped to the description field of the user in the Vidyo portal. In order to stay consistent, they build an Authentication Processing Filter that modifies all multi-valued attributes into a single semicolon-seperated string attribute value. The code for this is as follows:

```
<?php
/**
 * Flat Attributes Authentication Processing filter
 *
 * Replace all multi-valued attributes with semicolon separated single-valued attribute.
 *
 * @package switch
 */
class sspmod_switch_Auth_Process_FlatAttributes extends SimpleSAML_Auth_ProcessingFilter
{

    /**
     * Process a authentication response
     *
     * This function modifies the multi-valued attributes
     *
     * @param array &$state The state of the response.
     *
     * @return void
     */
    public function process(&$state)
    {
        assert('is_array($state)');
        $attributes  = $state['Attributes'];

        // Loop through all attributes
        foreach ($attributes AS $attrkey => $attrval) {
           if (sizeof($attrval) > 1) {
               $str = '';
               foreach($attrval AS $singlevalue) {
                  $str .= $singlevalue . ';';
               }
               $state['Attributes'][$attrkey] = array(substr($str,0,-1));
            }
        }
    }
}
```

It is configured in config.php with the following line:

```
80 => 'switch:FlatAttributes',
```

Extension number provisioning

Unfortunately the extension numbers provisioned by Vidyo on first login has variable length (4 or 5 digits). This makes dialplans and services like NRENUM hard to deploy. If you need to assign extension numbers from a fixed range, you have to implement custom extension number provisioning, and pass on the extension number from the proxy to Vidyo.

NIIF has implemented an extension number provisioning module that gets a federated identifier as an argument and returns either the previous number assigned to that ID or a fresh extension number, or an error signal. (One possible error is that the number range is exhausted. Therefore the code must be protected from unauthorized access to avoid DoS.)

The provisioning module is hooked into the SimpleSAMLphp proxy by using the more generic attribute from REST API module.