

---

# **Authentic2 Documentation**

***Release 1.9.2***

**Mikaël Ates**

October 20, 2011



# CONTENTS

<b>1</b>	<b>Documentation content</b>	<b>3</b>
1.1	Features . . . . .	3
1.2	Download . . . . .	3
1.3	Installation . . . . .	3
1.4	Authentication with an existing LDAP directory . . . . .	5
1.5	Authentication on Authentic2 with PAM . . . . .	6
1.6	How global policies are used in Authentic2 administration . . . . .	6
1.7	Where do I find the Authentic2 SAML2 metadata? . . . . .	7
1.8	Configure SAML 2.0 service providers . . . . .	7
1.9	Configure Authentic2 as a SAML2 service provider or a SAML2 proxy . . . . .	14
1.10	How to create/import and delete in bulk SAML2 identity and service providers with the sync-metadata script? . . . . .	21
1.11	Configure Authentic2 as a CAS server . . . . .	22
1.12	Configure Authentic2 as a CAS client . . . . .	22
1.13	Attribute Management in Authentic2 . . . . .	22
<b>2</b>	<b>Copyright</b>	<b>39</b>



Authentic2 is a versatile identity provider addressing a broad range of needs, from simple to advanced setups, around web authentication, attribute sharing, namespace mapping and authorization management.

Authentic2 supports many protocols and standards, including SAML2, CAS, OpenID, LDAP, X509, OATH, and can bridge between them.

Authentic2 is under the GNU AGPL version 3 licence.

It has support for SAMLv2 thanks to [Lasso](#), a free (GNU GPL) implementation of the Liberty Alliance and OASIS specifications of SAML2, ID-FF1.2 and ID-WSF2.

- [Authentic2 project site](#)
- [Authentic2 roadmap](#)



# DOCUMENTATION CONTENT

## 1.1 Features

Authentic can authenticate users against:

- an LDAP directory,
- a SAML 2.0 identity provider,
- an OpenID identity provider,
- with an X509 certificate.

Authentic can provide authentication to web applications using the following protocols:

- OpenID,
- SAML 2.0,
- CAS 1.0 & CAS 2.0.

Authentic can proxy authentication between any two different protocols it support.

## 1.2 Download

1. Pypi: <http://pypi.python.org/pypi/authentic2/1.9.0>
2. Git repository: <http://repos.entrouvert.org/authentic.git>
3. [Browse source](#)

## 1.3 Installation

### 1.3.1 Dependencies

You must install the following packages to use Authentic

- Python Lasso binding 2.3.5:  
From sources: <http://lasso.entrouvert.org/download> Debian based distribution: apt-get install python-lasso
- Django 1.3:

From sources: <http://www.djangoproject.com/download/1.3/tarball/>

- Django-registration 0.8-alpha-1:

From sources: <http://bitbucket.org/ubernostrum/django-registration/downloads> Debian based distribution: `apt-get install python-django-registration`

- Django-authopenid 0.9.6:

From sources: <http://bitbucket.org/benoitc/django-authopenid/downloads>

- Django-south 0.7.3:

From sources: <http://south.aeracode.org/docs/installation.html>

- Django-profiles 0.2:

From sources: <http://pypi.python.org/pypi/django-profiles>

You install all the django libraries quickly using `pip`:

```
pip install django django-profiles django-registration \
    django-debug-toolbar django-authopenid south
```

or `easy_install`:

```
easy_install django django-profiles django-registration \
    django-debug-toolbar django-authopenid south
```

### 1.3.2 Quick Start

Then launch the following commands:

```
python manage.py syncdb --migrate
python manage.py runserver
```

You should see the following output:

```
Validating models...
0 errors found
```

```
Django version 1.2, using settings 'authentic.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

You can access the running application on <http://127.0.0.1:8000/>

### 1.3.3 Specifying a different database

This is done by modifying the `DATABASES` dictionary in your `local_settings.py` file (create it in Authentic project directory); for example:

```
DATABASES['default'] = {
    'ENGINE': 'django.db.backends.postgresql',
    'NAME': 'authentic',
    'USER': 'admindb',
    'PASSWORD': 'foobar',
    'HOST': 'db.example.com',
    'PORT': '', # empty string means default value
}
```



You should refer to the Django documentation on databases settings at <http://docs.djangoproject.com/en/dev/ref/settings/#databases> for all the details.

### 1.3.4 How to upgrade to a new version of authentic ?

Authentic store all its data in a relational database as specified in its settings.py or local\_settings.py file. So in order to upgrade to a new version of authentic you have to update your database schema using the migration command — you will need to have installed the dependency django-south, see the beginning of this README file.:

```
python ./manage.py migrate
```

Then you will need to create new tables if there are.:

```
python ./manage.py syncdb
```

## 1.4 Authentication with an existing LDAP directory

Authentic use the module django\_auth\_ldap to synchronize the Django user tables with an LDAP. For complex use case, we will refer you to the django\_auth\_ldap documentation, see <http://packages.python.org/django-auth-ldap/>.

### 1.4.1 How to authenticate users against an LDAP server with anonymous binding ?

1. Install the django\_auth\_ldap module for Django:

```
pip install django_auth_ldap
```

2. Configure your local\_settings.py file for authenticating against LDAP.

The next lines must be added:

```
AUTHENTICATION_BACKENDS += ( 'django_auth_ldap.backend.LDAPBackend', )

import ldap
from django_auth_ldap.config import LDAPSearch

# Here put the LDAP URL of your server
AUTH_LDAP_SERVER_URI = 'ldap://ldap.example.com'
# Let the bind DN and bind password blank for anonymous binding
AUTH_LDAP_BIND_DN = ""
AUTH_LDAP_BIND_PASSWORD = ""
# Lookup user under the branch o=base and by matching their uid against the
# received login name
AUTH_LDAP_USER_SEARCH = LDAPSearch("o=base",
    ldap.SCOPE_SUBTREE, "(uid=%(user)s)")
```

### 1.4.2 How to allow members of an LDAP group to manage Authentic ?

1. First you must know the objectClass of groups in your LDAP schema, this FAQ will show you the configuration for two usual classes: groupOfNames and groupOfUniqueNames.
2. Find the relevant groupname. We will say it is: cn=admin,o=mycompany

3. Add the following lines:

```
from django_auth_ldap.config import GroupOfNamesType
AUTH_LDAP_GROUP_TYPE = GroupOfNamesType()
AUTH_LDAP_GROUP_SEARCH = LDAPSearch("o=mycompany",
    ldap.SCOPE_SUBTREE, "(objectClass=groupOfNames)")
AUTH_LDAP_USER_FLAGS_BY_GROUP = {
    "is_staff": "cn=admin,o=mycompany"
}
```

For an objectClass of groupOfUniqueNames you would change the string GroupOfNamesType to GroupOfUniqueNamesType and grouOfNames to groupOfUniqueNames. For more complex cases see the django\_auth\_ldap documentation.

## 1.5 Authentication on Authentic2 with PAM

This module is copied from <https://bitbucket.org/wnielson/django-pam/> by Weston Nielson and the pam ctype module by Chris Atlee <http://atlee.ca/software/pam/>.

Add 'authentic2.vendor.dpam.backends.PAMBackend' to your settings.py:

```
AUTHENTICATION_BACKENDS = (
    ...
    'authentic2.vendor.dpam.backends.PAMBackend',
    ...
)
```

Now you can login via the system-login credentials. If the user is successfully authenticated but has never logged-in before, a new User object is created. By default this new User has both is\_staff and is\_superuser set to False. You can change this behavior by adding PAM\_IS\_STAFF=True and PAM\_IS\_SUPERUSER in your settings.py file.

The default PAM service used is login but you can change it by setting the PAM\_SERVICE variable in your settings.py file.

## 1.6 How global policies are used in Authentic2 administration

The policy management with global policies is nearly used for any kind of policy in Authentic2.

For each kind of these policies, the system takes in account two special global policies named 'Default' and 'All':

- If no other policy applies, the policy 'Default' will apply.
- A policy can be created and attached to any related object. This policy is authoritative on policy 'Default'.
- If the policy 'All' exists, it is authoritative on any other policy.
- The global policies must be created by the administrator if necessary.

**A policy is taken in account only if it is enabled.**

```
def get_sample_policy(any_object):
    try:
        return SamplePolicy.objects.get(name='All', enabled=True)
    except SamplePolicy.DoesNotExist:
        pass
    if any_object.enable_following_sample_policy:
```

```

    if any_object.sample_policy:
        return any_object.sample_policy
    try:
        return SamplePolicy.objects.get(name='Default', enabled=True)
    except SamplePolicy.DoesNotExist:
        pass
    return None

```

*It is advised to add a 'Default' global policy when it is expected to apply a policy to all related objects. Add e regular policy to some objects are then used to handle particular configurations.*

*A 'Default' global policy should be defined to avoid misonfiguration.*

*A 'All' global policy should be used to enforce a global configuration for all related objects or for testing purposes.*

## 1.7 Where do I find the Authentic2 SAML2 metadata?

The SAML2 metadata are automatically generated.

**Authentic2 will infer from environment variables the host and port to generate the URLs contained in the metadata.**

The metadata of Authentic2 SAML2 identity provider are available at:

`http[s]://your.domain.com/idp/saml2/metadata`

The metadata of Authentic2 SAML2 service provider are available at:

`http[s]://your.domain.com/authsaml2/metadata`

## 1.8 Configure SAML 2.0 service providers

### 1.8.1 How do I authenticate against Authentic2 with a SAML2 service provider?

1. Declare Authentic2 as a SAML2 identity provider on your SAML2 service provider using the SAML2 identity provider metadata of Authentic2.

Go to `http[s]://your.domain.com/idp/saml2/metadata`

2. Add and configure a SAML2 service provider in Authentic2 using the metadata of the service provider.

### 1.8.2 How do I add and configure a SAML2 service provider in Authentic2?

You first need to create a new SAML2 service provider entry. This requires the SAML2 metadata of the service provider.

If your service provider is Authentic2, the metadata are available at:

`http[s]://your.domain.com/authsaml2/metadata`

See [Where do I find the Authentic2 SAML2 metadata?](#) for more information.

## Create a SAML2 service provider entry

1. Go to  
`http[s]://your.domain.com/admin/saml/libertyprovider/add/`
2. Fill the form fields

Authentic administration
Welcome, **mikael**. Change password / Log out

Home > Saml > Liberty providers > Add liberty provider

### Add liberty provider

Name:	<input type="text" value="My_service_provider"/>	<small>Internal nickname for the service provider</small>
Entity id:		
Entity id sha1:		
Federation source:	(None)	

Metadata files

Metadata:	<input type="text" value="/Donnees/Donnees/devs/authentic"/> <input type="button" value="Parcourir..."/>
-----------	--

Liberty Service Providers

Liberty Service Provider: #1

<input checked="" type="checkbox"/> Enabled
<input type="checkbox"/> The following options policy will apply except if a policy for all identity provider is defined.
SP Options Policy: <input type="text" value="-----"/> <input type="button" value="+"/>
Protocol policy: <input type="text" value="Default (AuthnRequest signature: Let authentic decides which signatures to check)"/> <input type="button" value="+"/>
Attribute policy: <input type="text" value="-----"/> <input type="button" value="+"/>

Liberty Identity Providers

Liberty Identity Provider: #1

<input type="checkbox"/> Enabled
<input type="checkbox"/> The following options policy will apply except if a policy for all identity provider is defined.
IdP Options Policy: <input type="text" value="-----"/> <input type="button" value="+"/>
<input type="checkbox"/> The following authorization policy will apply except if a policy for all identity provider is defined.
Authorization Policy: <input type="text" value="-----"/> <input type="button" value="+"/>

The service provider must be enabled.

See below about configuring the service provider with policies:

- options of the service provider
- protocol policy
- attribute policy

### 3. Save

**Authentic administration** Welcome, **mikael**. [Change password](#) / [Log out](#)

[Home](#) > [Saml](#) > [Liberty providers](#)

✓ The liberty provider "My\_service\_provider" was added successfully.

#### Select liberty provider to change Add liberty provider +

Action:   0 of 1 selected

<input type="checkbox"/>	Name	Entity id	Protocol conformance
<input type="checkbox"/>	My_service_provider	<a href="http://sp.mik.lan:8000/authsaml2/metadata">http://sp.mik.lan:8000/authsaml2/metadata</a>	SAML 2.0

1 liberty provider

## Configure the SAML2 service provider options

The SAML2 options of the service provider are configured using sp options policies.

See the *administration with policy principle* page [How global policies are used in Authentic2 administration](#).

You may create a regular policy and configure your service provider to use it.

Go to:

`http[s]://your.domain.com/admin/saml/spoptionsidppolicy/add/`

## Add identity provider options policy

<b>Nom:</b>	<input type="text" value="idp_policy_1"/>		
<input checked="" type="checkbox"/> Enabled			
<input type="checkbox"/> Do not send a namelD Policy			
<b>Requested name id format:</b>	<input type="text" value="Persistent"/>		
<input type="checkbox"/> This IdP falsely sends a transient NameID which is in fact persistent			
<input checked="" type="checkbox"/> Allow IdP to create an identity			
<input checked="" type="checkbox"/> Binding for Authnresponse (taken from metadata by the IdP if not enabled)	:	<input type="text" value="POST binding"/>	
<input checked="" type="checkbox"/> HTTP method for single logout request (taken from metadata if not enabled)	:	<input type="text" value="SOAP binding"/>	
<input checked="" type="checkbox"/> HTTP method for federation termination request (taken from metadata if not enabled)	:	<input type="text" value="SOAP binding"/>	
<b>Ask user consent:</b>	<input type="text" value="Implicit"/>		
<input type="checkbox"/> Force authentication			
<input type="checkbox"/> Passive authentication			
<input type="checkbox"/> Want AuthnRequest signed			
<b>Behavior with persistent namelD:</b>	<input type="text" value="Account linking by authentication"/>		
<b>Behavior with transient namelD:</b>	<input type="text" value="Ask authentication"/>		
<b>Return URL after a successful authentication:</b>	<input type="text" value="/"/>		
<input type="button" value="Save and add another"/> <input type="button" value="Save and continue editing"/> <input type="button" value="Save"/>			

Liberty Identity Providers	
<b>Liberty Identity Provider: LibertyIdentityProvider object</b>	<input type="checkbox"/> Delete
<input checked="" type="checkbox"/> Enabled	
<input type="checkbox"/> The following options policy will apply except if a policy for all identity provider is defined.	
IdP Options Policy:	<input type="text" value="idp_policy_1"/> <input type="button" value="Add"/>
<input type="button" value="Delete"/> <input type="button" value="Save and add another"/> <input type="button" value="Save and continue editing"/> <input type="button" value="Save"/>	

Exemple with a policy 'Default':

## Authentic administration

Welcome, **mikael**. [Change password](#) / [Log out](#)[Home](#) > [Saml](#) > [Identity provider options policies](#) > [Add identity provider options policy](#)

## Add identity provider options policy

<b>Nom:</b>	<input type="text" value="Default"/>		
<input checked="" type="checkbox"/> Enabled			
<input type="checkbox"/> Do not send a namelid Policy			
<b>Requested name id format:</b>	<input type="text" value="Persistent"/>		
<input type="checkbox"/> This IdP falsely sends a transient NamelID which is in fact persistent			
<input checked="" type="checkbox"/> Allow IdP to create an identity			
<input type="checkbox"/> Binding for Authnresponse (taken from metadata by the IdP if not enabled)	:	<input type="text" value="POST binding"/>	
<input type="checkbox"/> HTTP method for single logout request (taken from metadata if not enabled)	:	<input type="text" value="SOAP binding"/>	
<input type="checkbox"/> HTTP method for federation termination request (taken from metadata if not enabled)	:	<input type="text" value="SOAP binding"/>	
<b>Ask user consent:</b>	<input type="text" value="Implicit"/>		
<input type="checkbox"/> Force authentication			
<input type="checkbox"/> Passive authentication			
<input type="checkbox"/> Want AuthnRequest signed			
<b>Behavior with persistent namelid:</b>	<input type="text" value="Account linking by authentication"/>		
<b>Behavior with transient namelid:</b>	<input type="text" value="Ask authentication"/>		
<b>Return URL after a successful authentication:</b>	<input type="text" value="/"/>		
<input type="button" value="Save and add another"/> <input type="button" value="Save and continue editing"/> <input type="button" value="Save"/>			

## Authentic administration

Welcome, **mikael**. [Change password](#) / [Log out](#)[Home](#) > [Saml](#) > [Service provider options policies](#)

The service provider options policy "Default" was added successfully.

## Select service provider options policy to change

[Add service provider options policy](#)

Action:	<input type="text" value="-----"/>	<input type="button" value="Go"/>	0 of 1 selected
<input type="checkbox"/>	Service provider options policy		
<input type="checkbox"/>	Default		
1 service provider options policy			

Exemple with a policy 'All':

Authentic administration
Welcome, **mikael**. Change password / Log out

Home > Saml > Identity provider options policies > All

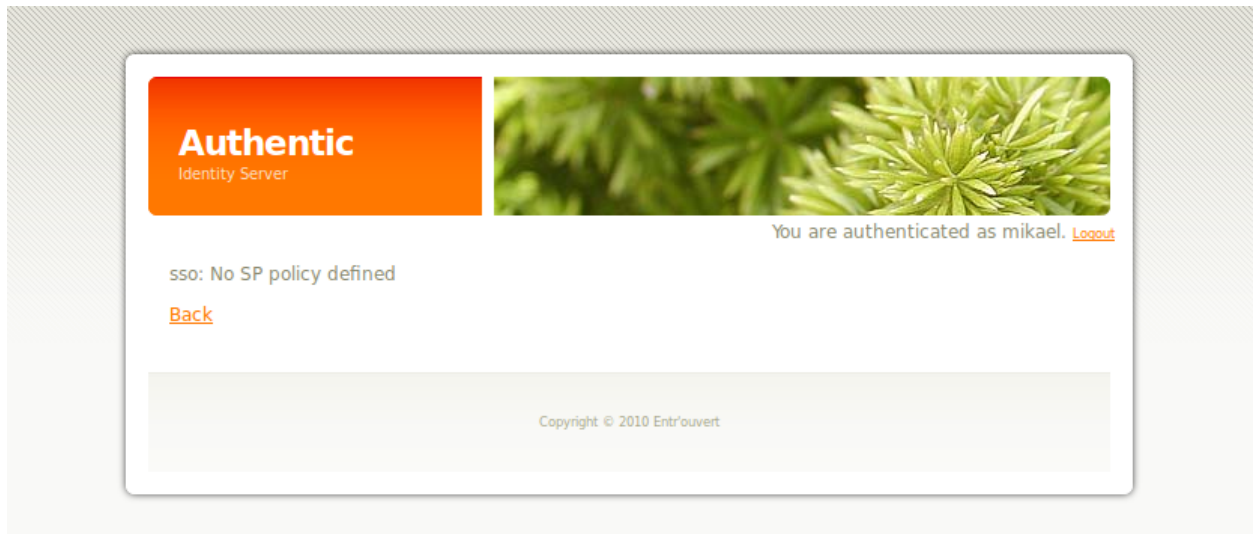
### Change identity provider options policy History

<b>Nom:</b>	<input type="text" value="All"/>		
<input checked="" type="checkbox"/> Enabled			
<input type="checkbox"/> Do not send a namelD Policy			
<b>Requested name id format:</b>	<input type="text" value="Persistent"/>		
<input type="checkbox"/> This IdP falsely sends a transient NameID which is in fact persistent			
<input checked="" type="checkbox"/> Allow IdP to create an identity			
<input checked="" type="checkbox"/> Binding for Authnresponse (taken from metadata by the IdP if not enabled)	:	<input type="text" value="Artifact binding"/>	
<input checked="" type="checkbox"/> HTTP method for single logout request (taken from metadata if not enabled)	:	<input type="text" value="Redirect binding"/>	
<input checked="" type="checkbox"/> HTTP method for federation termination request (taken from metadata if not enabled)	:	<input type="text" value="Redirect binding"/>	
<b>Ask user consent:</b>	<input type="text" value="Implicit"/>		
<input type="checkbox"/> Force authentication			
<input type="checkbox"/> Passive authentication			
<input type="checkbox"/> Want AuthnRequest signed			
<b>Behavior with persistent namelD:</b>	<input type="text" value="Account linking by authentication"/>		
<b>Behavior with transient namelD:</b>	<input type="text" value="Ask authentication"/>		
<b>Return URL after a successful authentication:</b>	<input type="text" value="/"/>		

✖ Delete
Save and add another
Save and continue editing
Save

If no policy is found for the configuration of the SAML2 options of a service provider, the following error is displayed when a SSO request is received.





### Configure the SAML2 service provider protocol options

This kind of policy does not use the policy management using global policies.

You should use the default option except if your service provider is a Shibboleth service provider.

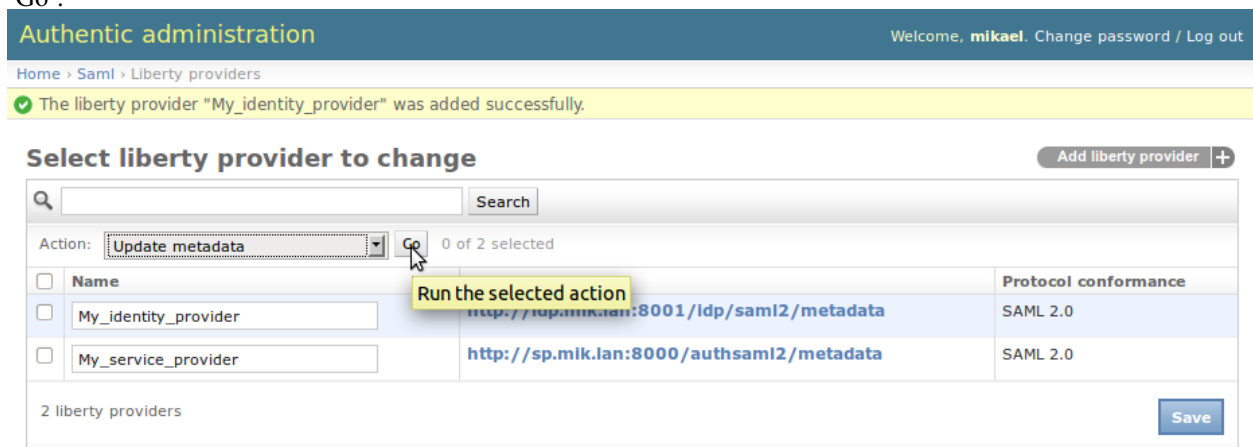
### Configure the attribute policy of the service provider

See the attribute management page *Attribute Management in Authentic2*.

### How to refresh metadata of an identity provider hosted at a Well-Known Location?

The Well-Known Location (WKL) means that the entity Id of the provider is a URL at which the provider metadata are hosted.

To refresh them, select the provider on the list of provider, then select in the menu 'Update metadata', then click on 'Go'.



## How to create in bulk service providers with the sync-metadata script?

See the page explaining the use of the script sync-metadata *How to create/import and delete in bulk SAML2 identity and service providers with the sync-metadata script?*.

## 1.9 Configure Authentic2 as a SAML2 service provider or a SAML2 proxy

The configuration to make Authentic2 a SAML2 service provider or a SAML2 proxy is the same. The difference comes from that Authentic2 is may be configured or not as a SAML2 identity provider.

### 1.9.1 How do I authenticate against a third SAML2 identity provider?

1. Declare Authentic2 as a SAML2 service provider on your SAML2 identity provider using the SAML2 service provider metadata of Authentic2.

Go to `http[s]://your.domain.com/authsaml2/metadata`

2. Add and configure a SAML2 identity provider entry in Authentic2 using the metadata of the identity provider.

### 1.9.2 How do I add and configure a SAML2 identity provider in Authentic2?

You first need to create a SAML2 identity provider entry with the SAML2 metadata of the identity provider. Then, you configure it.

If your identity provider is Authentic2, the metadata are available at:

`http[s]://your.domain.com/idp/saml2/metadata`

See *Where do I find the Authentic2 SAML2 metadata?* for more information.

### Create a SAML2 identity provider entry

You first need to create a new SAML2 identity provider entry. This requires the SAML2 metadata of the identity provider.

1. Go to  
`http[s]://your.domain.com/admin/saml/libertyprovider/add/`
2. Fill the form fields

Authentic administration Welcome, **mikael**. [Change password](#) / [Log out](#)

Home > Saml > Liberty providers > Add liberty provider

### Add liberty provider

Name:   
Internal nickname for the service provider

Entity id:

Entity id sha1:

Federation source: (None)

Metadata files

Metadata:

[Parcourir...](#)

Liberty Identity Providers

Liberty Identity Provider: #1

☒ Enabled

☐ The following options policy will apply except if a policy for all identity provider is defined.

IdP Options Policy:  [+](#)

[Save and add another](#) [Save and continue editing](#) [Save](#)

The identity provider must be enabled.


See below about configuring the identity provider with policies:

- options of the identity provider

### 3. Save

Authentic administration Welcome, **mikael**. [Change password](#) / [Log out](#)

Home > Saml > Liberty providers

 The liberty provider "My\_identity\_provider" was added successfully.

### Select liberty provider to change Add liberty provider +

[Search](#)

Action:  [Go](#) 0 of 2 selected

<input type="checkbox"/>	Name	Entity id	Protocol conformance
<input type="checkbox"/>	<input type="text" value="My_identity_provider"/>	<a href="http://ldp.mik.ian:8001/ldp/saml2/metadata">http://ldp.mik.ian:8001/ldp/saml2/metadata</a>	SAML 2.0
<input type="checkbox"/>	<input type="text" value="My_service_provider"/>	<a href="http://sp.mik.ian:8000/authsaml2/metadata">http://sp.mik.ian:8000/authsaml2/metadata</a>	SAML 2.0

2 liberty providers [Save](#)

## Configure the SAML2 identity provider options

The SAML2 options of the service provider are configured using sp options policies.

See the *administration with policy principle* page [How global policies are used in Authentic2 administration](#).

You may create a regular policy and configure your service provider to use it.

Go to:

[http\[s\]://your.domain.com/admin/saml/idpoptionssppolicy/add/](http[s]://your.domain.com/admin/saml/idpoptionssppolicy/add/)

Authentic2 administration
Welcome, **mikael**. Change password / Log out

Home > Saml > Identity provider options policies > Add identity provider options policy

### Add identity provider options policy

<b>Nom:</b>	<input type="text" value="idp_policy_1"/>		
<input checked="" type="checkbox"/> Enabled			
<input type="checkbox"/> Do not send a namelD Policy			
<b>Requested name id format:</b>	<input type="text" value="Persistent"/>		
<input type="checkbox"/> This IdP falsely sends a transient NameID which is in fact persistent			
<input checked="" type="checkbox"/> Allow IdP to create an identity			
<input checked="" type="checkbox"/> Binding for Authnresponse (taken from metadata by the IdP if not enabled)	:	<input type="text" value="POST binding"/>	
<input checked="" type="checkbox"/> HTTP method for single logout request (taken from metadata if not enabled)	:	<input type="text" value="SOAP binding"/>	
<input checked="" type="checkbox"/> HTTP method for federation termination request (taken from metadata if not enabled)	:	<input type="text" value="SOAP binding"/>	
<b>Ask user consent:</b>	<input type="text" value="Implicit"/>		
<input type="checkbox"/> Force authentication			
<input type="checkbox"/> Passive authentication			
<input type="checkbox"/> Want AuthnRequest signed			
<b>Behavior with persistent namelD:</b>	<input type="text" value="Account linking by authentication"/>		
<b>Behavior with transient namelD:</b>	<input type="text" value="Ask authentication"/>		
<b>Return URL after a successful authentication:</b>	<input type="text" value="/"/>		

**Authentic administration**Welcome, **mikael**. [Change password](#) / [Log out](#)

[Home](#) > [Saml](#) > [Identity provider options policies](#)

✓ The identity provider options policy "idp\_policy\_1" was added successfully.

### Select identity provider options policy to change

Add identity provider options policy +

Action:   0 of 1 selected

<input type="checkbox"/>	Identity provider options policy
<input type="checkbox"/>	<b>idp_policy_1</b>

1 Identity provider options policy

**Liberty Identity Providers**

**Liberty Identity Provider: LibertyIdentityProvider object** ☐ Delete

☒ Enabled

☐ The following options policy will apply except if a policy for all identity provider is defined.

IdP Options Policy:  +

Exemple with a policy 'Default':

Authentic administration Welcome, **mikael**. [Change password](#) / [Log out](#)

Home > SAML > Identity provider options policies > Add identity provider options policy

### Add identity provider options policy

<b>Nom:</b>	<input type="text" value="Default"/>		
<input checked="" type="checkbox"/> Enabled			
<input type="checkbox"/> Do not send a namelid Policy			
<b>Requested name id format:</b>	<input type="text" value="Persistent"/>		
<input type="checkbox"/> This IdP falsely sends a transient NameID which is in fact persistent			
<input checked="" type="checkbox"/> Allow IdP to create an identity			
<input type="checkbox"/> Binding for Authnresponse (taken from metadata by the IdP if not enabled)	:	<input type="text" value="POST binding"/>	
<input type="checkbox"/> HTTP method for single logout request (taken from metadata if not enabled)	:	<input type="text" value="SOAP binding"/>	
<input type="checkbox"/> HTTP method for federation termination request (taken from metadata if not enabled)	:	<input type="text" value="SOAP binding"/>	
<b>Ask user consent:</b>	<input type="text" value="Implicit"/>		
<input type="checkbox"/> Force authentication			
<input type="checkbox"/> Passive authentication			
<input type="checkbox"/> Want AuthnRequest signed			
<b>Behavior with persistent namelid:</b>	<input type="text" value="Account linking by authentication"/>		
<b>Behavior with transient namelid:</b>	<input type="text" value="Ask authentication"/>		
<b>Return URL after a successful authentication:</b>	<input type="text" value="/"/>		

Exemple with a policy ‘All’:

Authentic administration
Welcome, **mikael**. Change password / Log out

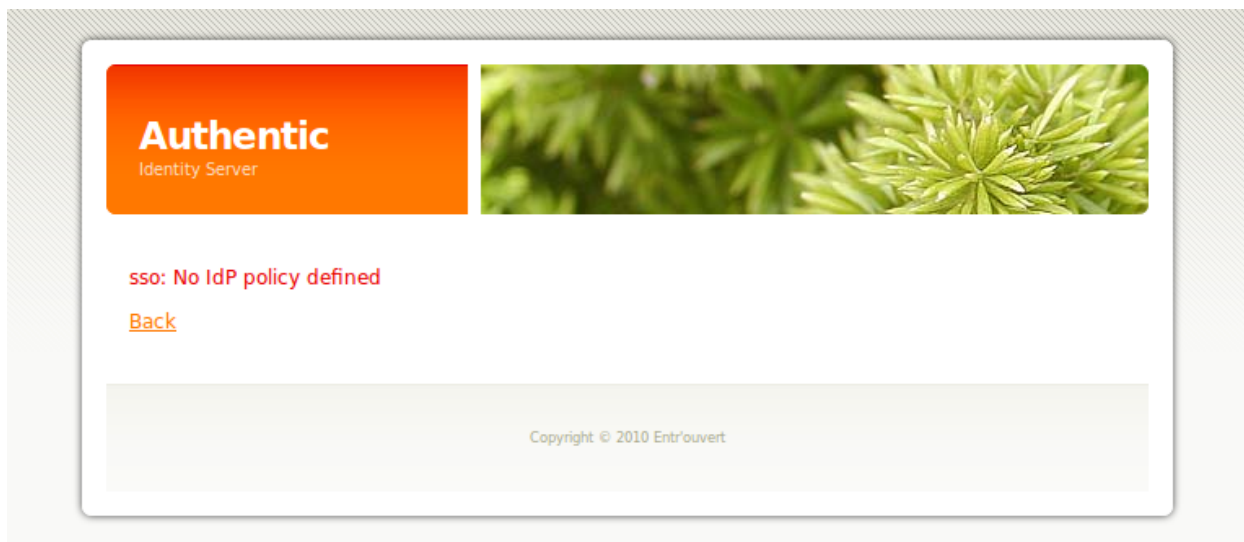
Home > Saml > Identity provider options policies > All

### Change identity provider options policy History

<b>Nom:</b>	All
<input checked="" type="checkbox"/> Enabled	
<input type="checkbox"/> Do not send a namelid Policy	
<b>Requested name id format:</b>	Persistent
<input type="checkbox"/> This IdP falsely sends a transient NameID which is in fact persistent	
<input checked="" type="checkbox"/> Allow IdP to create an identity	
<input checked="" type="checkbox"/> Binding for Authnresponse (taken from metadata by the IdP if not enabled)	: Artifact binding
<input checked="" type="checkbox"/> HTTP method for single logout request (taken from metadata if not enabled)	: Redirect binding
<input checked="" type="checkbox"/> HTTP method for federation termination request (taken from metadata if not enabled)	: Redirect binding
<b>Ask user consent:</b>	Implicit
<input type="checkbox"/> Force authentication	
<input type="checkbox"/> Passive authentication	
<input type="checkbox"/> Want AuthnRequest signed	
<b>Behavior with persistent namelid:</b>	Account linking by authentication
<b>Behavior with transient namelid:</b>	Ask authentication
<b>Return URL after a successful authentication:</b>	/

✖ Delete
Save and add another
Save and continue editing
Save

If no policy is found for the configuration of the SAML2 options of an identity provider, the following error is displayed when a SSO request is initiated.



## How to refresh metadata of an identity provider hosted at a Well-Known Location?

The Well-Known Location (WKL) means that the entity Id of the provider is a URL at which the provider metadata are hosted.

To refresh them, select the provider on the list of provider, then select in the menu 'Update metadata', then click on 'Go'.

Authentic administration Welcome, **mikael**. [Change password](#) / [Log out](#)

[Home](#) > [Saml](#) > [Liberty providers](#)

✓ The liberty provider "My\_identity\_provider" was added successfully.

### Select liberty provider to change Add liberty provider +

Search:

Action: Update metadata  0 of 2 selected

<input type="checkbox"/>	Name	Entity id	Protocol conformance
<input type="checkbox"/>	My_identity_provider	<a href="http://idp.mik.lan:8001/ldp/saml2/metadata">http://idp.mik.lan:8001/ldp/saml2/metadata</a>	SAML 2.0
<input type="checkbox"/>	My_service_provider	<a href="http://sp.mik.lan:8000/authsaml2/metadata">http://sp.mik.lan:8000/authsaml2/metadata</a>	SAML 2.0

2 liberty providers

Authentic administration Welcome, **mikael**. [Change password](#) / [Log out](#)

[Home](#) > [Saml](#) > [Liberty providers](#)

✓ Metadata update for: <http://idp.mik.lan:8001/ldp/saml2/metadata>

### Select liberty provider to change Add liberty provider +

Search:

Action: -----  0 of 2 selected

<input type="checkbox"/>	Name	Entity id	Protocol conformance
<input type="checkbox"/>	My_identity_provider	<a href="http://idp.mik.lan:8001/ldp/saml2/metadata">http://idp.mik.lan:8001/ldp/saml2/metadata</a>	SAML 2.0
<input type="checkbox"/>	My_service_provider	<a href="http://sp.mik.lan:8000/authsaml2/metadata">http://sp.mik.lan:8000/authsaml2/metadata</a>	SAML 2.0

2 liberty providers



## How to create in bulk identity providers with the sync-metadata script?

See the page explaining the use of the script sync-metadata *How to create/import and delete in bulk SAML2 identity and service providers with the sync-metadata script?*.

## 1.10 How to create/import and delete in bulk SAML2 identity and service providers with the sync-metadata script?

This section explains how to use the script sync-metadata.

### 1.10.1 Presentation

This script allows to create/import and delete in bulk SAML2 identity and service providers using standard SAML2 metadata files containing entity descriptors.

An example of such a file used in production is the global metadata file of the identity federation of French universities that can be found at <http://...>

Use the following command:

```
path_to_project/authentic2$ python manage.py sync-metadata file_name [options]
```

### 1.10.2 Options

- idp  
Load only identity providers of the metadata file.
- sp  
Load only service providers of the metadata file.
- source  
Used to tag all imported providers with a label. This option is used to metadata reloading and deletion in bulk.  
  
Reloading a metadata file, when a provider with same entity is found, it is updated. If a provider in the metadata file does not exist it is created. If a provider exists in the system but not in the metadata file, it is removed.

**For reloading, a source can only be associated with a unique metadata file. This is due to the fact that all providers of a source not found in the metadata file are removed.**

```
path_to_project/authentic2$ python manage.py sync-metadata file_name --source=french_federation
```

- sp-policy  
To configure the SAML2 parameters of service providers imported with the script, a policy of type `SPOptionsIdPPolicy` must be created in the administration interface. Either it is a global policy 'Default' or 'All' or it is a regular policy. If it is a regular policy, the policy name can be specified in parameter of the script with this option. The policy is then associated to all service providers created.

```
path_to_project/authentic2$ python manage.py sync-metadata file_name --sp-policy=sp_policy_name
```

- idp-policy

To configure the SAML2 parameters of identity providers imported with the script, a policy of type `IdPOptionsSPPolicy` must be created in the the administration interface. Either it is a global policy 'Default' or 'All' or it is a regular policy. If it is a regular policy, the policy name can be specified in parameter of the script with this option. The policy is then associated to all service providers created.

```
path_to_project/authentic2$ python manage.py sync-metadata file_name --idp-policy=idp_policy_name
```

- delete

With no options, all providers are deleted.

With the source option, only providers with the source name given are deleted.

**This option can not be combined with options `idp` and `sp`.**

- ignore-errors

If loading of one `EntityDescriptor` fails, continue loading

## 1.11 Configure Authentic2 as a CAS server

### 1.11.1 How to use Authentic2 as a CAS 1.0 or CAS 2.0 identity provider ?

1. Activate CAS IdP support in `settings.py`:

```
IDP_CAS = True
```

2. Then create the database table to hold CAS service tickets:

```
python authentic2/manage.py syncdb --migrate
```

3. Also configure authentic2 to authenticate against your LDAP directory (see above) if your want your user attributes to be accessible from your service, if it is not necessary you can use the normal relational database storage for you users.

4. Finally configure your service to point to the CAS endpoint at:

```
http[s]://your.domain.com/idp/cas/
```

5. If needed configure your service to resolve authenticated user with your LDAP directory (if user attributes are needed for your service)

## 1.12 Configure Authentic2 as a CAS client

## 1.13 Attribute Management in Authentic2

### 1.13.1 Summary

Attribute management currently allows to configure attribute policies associated with SAML2 service providers to define attributes that are pushed in SAML2 successful authentication response delivered by Authentic2.

User attributes can be taken from LDAP directories, the user Django profile or taken from the user Django session if Authentic2 is also configured as a SAML2 service provider.

Indeed, when Authentic2 acts also as a SAML2 service provider, attributes contained in the SAML2 assertion received from third IdP are put in the user session.

Attributes can thus be proxified during SSO with Authentic2 configured as a SAML2 proxy.

The namespace of attributes received from another SAML2 IdP or pushed in the assertion given in to service providers can be configured per attribute or per service provider.

By default, the namespace and format of attributes in assertion is conformant to the SAMLV2.0 X500/LDAP Attribute profile:

```
<saml:Attribute
  xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
  NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
  Name="urn:oid:2.5.4.42" FriendlyName="givenName">
  <saml:AttributeValue xsi:type="xs:string"
    x500:Encoding="LDAP">Mikaël</saml:AttributeValue>
</saml:Attribute>
```

But the <http://schemas.xmlsoap.org/ws/2005/05/identity/claims> from the ISI profile can also be used, for instance:

```
<saml:Attribute
  NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
  Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"
  FriendlyName="First Name">
  <saml:AttributeValue>Mikaël</saml:AttributeValue>
</saml:Attribute>
```

## 1.13.2 Configuration

### Configure sources of attributes

The source of attributes for authentic2 are of two kinds. The LDAP sources and the user django profile.

#### Declare the Django profile source

Add an attribute source named USER\_PROFILE with namespace 'Default'.

1. Go to [http\[s\]://your.domain.com/admin/attribute\\_aggregator/attributesource/add/](http[s]://your.domain.com/admin/attribute_aggregator/attributesource/add/)
2. Write 'USER\_PROFILE' in name field

Authentic administration Welcome, **mikaël**. [Change password](#) / [Log out](#)

[Home](#) > [Attribute aggregator](#) > [Attribute sources](#) > [Add attribute source](#)


### Add attribute source

<b>Name:</b>	<input type="text" value="USER_PROFILE"/>
<b>Namespace:</b>	<input type="text" value="Default"/>

3. Save

Authentic administration Welcome, **mikael**. [Change password](#) / [Log out](#)

Home > Attribute\_aggregator > Attribute sources

 The attribute source "USER\_PROFILE" was added successfully.

Select attribute source to change Add attribute source +

Action:   0 of 1 selected

<input type="checkbox"/>	Attribute source
<input type="checkbox"/>	USER_PROFILE

1 attribute source

## Add an LDAP Source

For LDAP sources, objects of type 'LDAPSource' must be created.

**Even if the authentication is based on LDAP authentication, thus that a server is configured in settings.py, it is necessary to create a corresponding 'LDAPSource' to use it as a source of attribute.**

1. Go to `http[s]://your.domain.com/admin/attribute_aggregator/ldapsource/add/`
2. Fill form fields

Only the field Name, Server, User, Password, Base and Port are used for now. **The namespace of LDAP source must be kept to 'Default', since the system namespace is based on LDAP.**

Authentic administration Welcome, **mikael**. [Change password](#) / [Log out](#)

Home > Attribute\_aggregator > Ldap sources > Add ldap source

Add ldap source

Name:

Namespace:

Server:

User:

Password:

Base:

Port:

☐ Ldaps

Certificate:

☐ Is auth backend

## 3. Save

Authentic administration Welcome, **mikael**. [Change password](#) / [Log out](#)

[Home](#) > [Attribute\\_aggregator](#) > [Ldap sources](#)

✓ The ldap source "LDAP Central" was added successfully.

**Select ldap source to change** Add ldap source +

Action:	Go	0 of 1 selected
<input type="checkbox"/> Ldap source		
<input checked="" type="checkbox"/> LDAP Central		

1 ldap source

## Manage user distinguished names in LDAP directories

To find the user in a LDAP directory, authentic2 must know its distinguished name (DN). If this LDAP has been used when the user has authenticated, Authentic2 learn the user DN. Nothing has to be done from this point of view.

However, if it is expected that user attributes be taken in a directory that is not used by the user for authentication, it is necessary to manually indicate to Authentic2 what is the user DN in the directory. For this, a user alias in source is created for the user:

1. Go to `http[s]://your.domain.com/admin/attribute_aggregator/useraliasinsource/add/`
2. Fill form fields

Authentic administration Welcome, **mikael**. [Change password](#) / [Log out](#)

[Home](#) > [Attribute\\_aggregator](#) > [Aliases in source](#) > [Add alias in source](#)

**Add alias in source**

**Name:**

**Attribute Source:**  +

**User:**  +

[Save and add another](#) [Save and continue editing](#) [Save](#)

## 3. Save

Authentic administration Welcome, **mikael**. [Change password](#) / [Log out](#)

[Home](#) > [Attribute\\_aggregator](#) > [Aliases in source](#)

✓ The alias in source "alias cn=mikael,ou=people,dc=entrouvert,dc=lan of user mikael in LDAP Central" was added successfully.

**Select alias in source to change** Add alias in source +

Action:	Go	0 of 1 selected
<input type="checkbox"/> Alias in source		
<input checked="" type="checkbox"/> alias cn=mikael,ou=people,dc=entrouvert,dc=lan of user mikael in LDAP Central		

1 alias in source

## Configure attributes pushed to SAML2 service providers in SSO response

Reminder:

- The default name format in SAML2 assertions is URI

- The default namespace called 'Default' is LDAP

In summary:

1. Create attribute items indicating an attribute name, a source, the name format expected and the namespace expected for the attribute name and friendly name if any.
2. Create a named list of attribute items.
3. Create an attribute policy and associate the previous list or associate the previous list to a existing attribute policy.
4. Associate the policy to a service provider.

### Create attribute items

1. Go to `http[s]://your.domain.com/admin/idp/attributeitem/add/`
2. Fill form fields

The screenshot shows the 'Add attribute of a list (SSO Login)' form. The form has the following fields:

- Attribute name:** A dropdown menu with 'sn' selected.
- Output name format:** A dropdown menu with 'SAMLv2 URI' selected.
- Output namespace:** A dropdown menu with 'Default' selected.
- Required:** An unchecked checkbox.
- Source:** A dropdown menu with 'LDAP Central' selected, followed by a green plus icon.

At the bottom of the form, there are three buttons: 'Save and add another', 'Save and continue editing', and 'Save'.

3. Save

The screenshot shows the 'Select attribute of a list (SSO Login) to change' page. The page has the following elements:

- Header:** 'Authentic administration' and 'Welcome, mikael. Change password / Log out'.
- Breadcrumbs:** 'Home > Idp > Attributes of lists (SSO Login)'.
- Success Message:** A green checkmark icon followed by the text: 'The attribute of a list (SSO Login) "sn (Output name format: urn:oasis:names:tc:SAML:2.0:attrname-format:uri) (Output namespace: Default) (Required: False) (Source: LDAP Central)" was added successfully.'
- Section Header:** 'Select attribute of a list (SSO Login) to change'.
- Buttons:** 'Add attribute of a list (SSO Login) +'.
- Action:** A dropdown menu with '-----' selected, followed by a 'Go' button and '0 of 1 selected'.
- Table:** A table with one row containing a checkbox and the text: 'sn (Output name format: urn:oasis:names:tc:SAML:2.0:attrname-format:uri) (Output namespace: Default) (Required: False) (Source: LDAP Central)'. The row is highlighted in blue.
- Footer:** '1 attribute of a list (SSO Login)'.

### Create a named list of attribute items

1. Go to `http[s]://your.domain.com/admin/idp/attributelist/add/`

## 2. Name the list and add items to list

**Authentic administration** Welcome, **mikael**. [Change password](#) / [Log out](#)

[Home](#) > [Idp](#) > [Attribute lists \(SSO Login\)](#) > [Add attribute list \(SSO Login\)](#)

### Add attribute list (SSO Login)

**Name:**

Attributes: Hold down "Control", or "Command" on a Mac, to select more than one.

**Available attributes**

**Chosen attributes**

Select your choice(s) and click

sn (Output name format: urn:oasis:names:tc:...

[Save and add another](#) [Save and continue editing](#) [Save](#)

## 3. Save

**Authentic administration** Welcome, **mikael**. [Change password](#) / [Log out](#)

[Home](#) > [Idp](#) > [Attribute lists \(SSO Login\)](#)

The attribute list (SSO Login) "Basic list 1" was added successfully.

### Select attribute list (SSO Login) to change

[Add attribute list \(SSO Login\)](#)

Action:   0 of 1 selected

<input type="checkbox"/>	Attribute list (SSO Login)
<input type="checkbox"/>	Basic list 1

1 attribute list (SSO Login)

## Create or modify an attribute policy

1. Go to [http\[s\]://your.domain.com/admin/idp/attributepolicy/add/](http[s]://your.domain.com/admin/idp/attributepolicy/add/)
2. Add list to the policy

**Authentic administration**Welcome, **mikael**. [Change password](#) / [Log out](#)

[Home](#) > [Idp](#) > [Attribute options policies](#) > [Add attribute options policy](#)

### Add attribute options policy

<b>Name:</b>	<input type="text" value="Basic policy 1"/>
Attribute list for sso from pull sources:	<div><div>Basic list 1</div><div>+</div></div>
<input type="checkbox"/> Forward attributes from push sources	
<input type="checkbox"/> Map attributes from push sources	
<b>Output name format:</b>	<div>SAMLv2 BASIC</div>
<b>Output namespace:</b>	<div>Default</div>
Source filter for sso from push sources:	<div><div>Hold down "Control", or "Command" on a Mac, to select more than one.</div><div><div><b>Available source filter for sso from push sources</b></div><div><input type="text" value=""/> USER_PROFILE http://idp2.mik.lan:8002/idp/saml2/metadata LDAP Central</div><div><div>Choose all</div></div></div><div><div><b>Chosen source filter for sso from push sources</b></div><div>Select your choice(s) and click +</div><div></div><div><div>Clear all</div></div></div></div>
Attribute filter for sso from push sources:	<div><div>-----</div><div>+</div></div>
<input type="checkbox"/> Filter source of filtered attributes	
<input type="checkbox"/> Map attributes of filtered attributes	
<input type="checkbox"/> Send error and no attrs if missing required attrs	

Save and add another

Save and continue editing

Save

## 3. Save



**Authentic administration** Welcome, **mikael**. [Change password](#) / [Log out](#)

[Home](#) > [Idp](#) > [Attribute options policies](#)

✓ The attribute options policy "Basic policy 1 not yet associated with a service provider" was added successfully.

### Select attribute options policy to change

[Add attribute options policy](#) +

Action:   0 of 1 selected

<input type="checkbox"/>	Attribute options policy
<input type="checkbox"/>	Basic policy 1 not yet associated with a service provider

1 attribute options policy

### Associate the policy to a service provider

1. Go to [http\[s\]://your.domain.com/admin/saml/libertyprovider/1/](http[s]://your.domain.com/admin/saml/libertyprovider/1/)
2. Add policy to the service provider

**Liberty Service Providers**

**Liberty Service Provider: #1**

☐ Enabled

**Preferred assertion consumer binding:**

☐ Encrypt NameID

☐ Encrypt Assertion

☐ AuthnRequest signed

☐ Allow IdP initiated SSO

**Default name id format:**

Accepted name id format: ☐ Aucun(e)

- ☐ Transient
- ☐ Persistent
- ☐ Email (only supported by SAMLv2)

☐ Ask user for consent when creating a federation

**Protocol policy:**  +

**Attribute policy:**  +

3. Save

**Authentic administration** Welcome, **mikael**. [Change password](#) / [Log out](#)

[Home](#) > [Saml](#) > [Liberty providers](#)

✓ The liberty provider "SP" was changed successfully.

### Select liberty provider to change

Add liberty provider +

Action:   0 of 2 selected

<input type="checkbox"/>	Name	Entity id	Protocol conformance
<input type="checkbox"/>	IdP2	<a href="http://idp2.mik.lan:8002/idp/saml2/metadata">http://idp2.mik.lan:8002/idp/saml2/metadata</a>	SAML 2.0
<input type="checkbox"/>	SP	<a href="http://sp.mik.lan:8000/authsaml2/metadata">http://sp.mik.lan:8000/authsaml2/metadata</a>	SAML 2.0

2 liberty providers

4. The display name of the policy has changed

**Authentic administration** Welcome, **mikael**. [Change password](#) / [Log out](#)

[Home](#) > [Idp](#) > [Attribute options policies](#)

### Select attribute options policy to change

Add attribute options policy +

0 of 1 selected

<input type="checkbox"/>	Attribute options policy
<input type="checkbox"/>	Basic policy 1 associated with SP

1 attribute options policy

## Handle attributes provided by other Identity providers, proxy attributes

Link to configure first Authentic as a sp to have attributes in session

Add a source if mapping set to true

### 1.13.3 Modifying supported namespaces and attribute name mappings

TBD

### 1.13.4 Explanation (Draft)

#### Attribute aggregator module

The core attribute management is based on the attribute aggregator module.

#### Intro

Attribute aggregator provides a main Model class called UserAttributeProfile, functions to load attributes and extract attributes.

The mapping between attribute namespaces is built-in and depends on a unique file (mapping.py).

A main schema is defined and is based on LDAP/X500 for naming. The support of <http://schemas.xmlsoap.org/ws/2005/05/identity/claims> is partly complete.

Source of attributes are connected with attribute loading functions using signals.

## FAQ

Why not use the Django User profile?

The django user profile needs to define attributes as class attributes and then support form filling or mapping with LDAP.

That is useful and may be used, especially because the profile can be used as a source of attribute to load in the attribute\_aggregator profile.

The attribute\_aggregator profile allow to load multivalued attributes from any source supported (LDAP, Django profile and attributes in Django session for now) from any namespace defined in mapping.py (LDAP/X500 and claims for now).

The profile can be loaded giving a source or a list of attribute, or can be from any known source, or with a dictionary.

Attributes can be extracted with many functions in any namespace supported.

## Quick explanation

The schema is defined in mapping.py and is made of definitions like this:

```
"sn": {
    "oid": "2.5.4.4",
    "display_name": _("sn surname"),
    "alias": ['surname'],
    "profile_field_name": 'last_name',
    "type": "http://www.w3.org/2001/XMLSchema#string",
    "namespaces": {
        "http://schemas.xmlsoap.org/ws/2005/05/identity/claims": {
            "identifiers":
                [
                    "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname",
                ],
            "friendly_names":
                [
                    "Last Name",
                ],
        }
    }
},
```

The profile store all the data in a text field taht contains a cPickle list of instances of the class AttributeData.

The profile is attached to a user and then can be created or loaded with:

```
profile = load_or_create_user_profile(user=user)
```

User may be None to create a temporary profile for an anonymous user. But that need a DB cleaning function not implemented.

### The model *UserAttributeProfile*

The model 'UserAttributeProfile' can be attached to a user and then persist (as a Model).

When the profile is loaded, all data stored are removed expect if the the data has an expiration date later.

The profile provide several methods to store and extract attributes.

All the methods to add attributes are based on a main one accepting a dictionary of attribute is parameters 'load\_by\_dic()'. The other methods ('load\_listed\_attributes()', 'load\_greedy()') send a signal with a list of attributes (listed\_attributes\_call) or not (any\_attributes\_call) to grab a dictionary. The list is given with the definition name, oid or friendly name of the attribute in the system namespace.

Into the dictionary, attributes are given with their name, oid or friendly name in the default namespace or with their name in a namespace. An expiration date can also be given (ISO8601 format), if none, attribute will be deleted at next profile loading. The dictionary format is as follows:

```
attributes = dict()
data_from_source = list()
a1 = dict()
    a1['oid'] = definition_name
Or
    a1['definition'] = definition_name
    definition may be the definition name like 'gn'
    or an alias like 'givenName'
Or
    a1['name'] = attribute_name_in_ns
    a1['namespace'] = ns_name
a1['expiration_date'] = date
a1['values'] = list_of_values
data_from_source.append(a1)
...
data_from_source.append(a2)
attributes[source_name] = data_from_source
```

Getters are defined to extract data from a profile. Only AttributeData instances are extracted that assume that any attribute namespace can be used.

- get\_data\_of\_definition(definition)

Return a list of AttributeData instances corresponding to the definition given.

- get\_freshest\_data\_of\_definition(definition)

Return the freshest AttributeData instance. If multiple with no or same exp date, random. Should use the creation date soon.

- get\_data\_of\_source

Return a list of AttributeData instances corresponding to the source given.

- get\_data\_of\_source\_by\_name

Idem but source name is given, not a Source instance.

- get\_data\_of\_definition\_and\_source

Return a list of AttributeData instances corresponding to the definition and source given.

- get\_data\_of\_definition\_and\_source\_by\_name

Idem but source name is given, not a Source instance.

## SAML2 attribute representation in assertions

SAML2 attribute profile (saml-profiles-2.0-os - Section 8) defines two kind of attribute element syntax in the attribute statement of assertions, also called *name format*:

- BASIC:

```
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
```

- URI:

```
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
```

*URI should be used when attributes have “universally” known unique names like OID.*

Example:

```
<saml:Attribute NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
  Name="FirstName">
  <saml:AttributeValue xsi:type="xs:string">By-Tor</saml:AttributeValue>
</saml:Attribute>

<saml:Attribute
  xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
  NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
  Name="urn:oid:2.5.4.42" FriendlyName="givenName">
  <saml:AttributeValue xsi:type="xs:string"
    x500:Encoding="LDAP">Steven</saml:AttributeValue>
</saml:Attribute>
```

## BASIC

Two <Attribute> elements refer to the same SAML attribute if and only if the values of their Name XML attributes are equal in the sense of Section 3.3.6 of [Schema2].

No additional XML attributes are defined for use with the <Attribute> element.

The schema type of the contents of the <AttributeValue> element MUST be drawn from one of the types defined in Section 3.3 of [Schema2]. The xsi:type attribute MUST be present and be given the appropriate value.

## X.500/LDAP Attribute Profile (URI)

### Extracted from the SAML2 core specifications

Two <Attribute> elements refer to the same SAML attribute if and only if their Name XML attribute values are equal in the sense of [RFC3061]. The FriendlyName attribute plays no role in the comparison.

Directory attribute type definitions for use in native X.500 directories specify the syntax of the attribute using ASN.1 [ASN.1]. For use in LDAP, directory attribute definitions additionally include an LDAP syntax which specifies how attribute or assertion values conforming to the syntax are to be represented when transferred in the LDAP protocol (known as an LDAP-specific encoding). The LDAP-specific encoding commonly produces Unicode characters in UTF-8 form. This SAML attribute profile specifies the form of SAML attribute values only for those directory attributes which have LDAP syntaxes. Future extensions to this profile may define attribute value formats for directory attributes whose syntaxes specify other encodings.

To represent the encoding rules in use for a particular attribute value, the <AttributeValue> element MUST contain an XML attribute named Encoding defined in the XML namespace urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500.

For any directory attribute with a syntax whose LDAP-specific encoding exclusively produces UTF-8 character strings as values, the SAML attribute value is encoded as simply the UTF-8 string itself, as the content of the <AttributeValue> element, with no additional whitespace. In such cases, the xsi:type XML attribute MUST be set to xs:string. The profile-specific Encoding XML attribute is provided, with a value of LDAP.

The AttributeData instances have a field expiration\_data. If the profile exists, obsolete data are removed at loading.

### When authentic 2 deals with attributes and needs mapping?

Authentic2 behaves as an attribute provider: \* At the SSO login \* When an attribute request is received

Authentic requests (e.g. by soap) are not yet supported.

### When Authentic2 behaves as an attribute provider at SSO login

At a SSO request, just before responding to the service provider, the saml2 idp module sends the signal 'add\_attributes\_to\_response' giving the SP entity ID.

The signal is connected to the function 'provide\_attributes\_at\_sso()' in charge of providing the attributes at the SSO for this SP.

**Attributes sources are of two kinds. The first ones are the sources that can be requested by the IdP with a synchronous binding without user interactions. These sources are called pull sources. They are for now limited to LDAP sources. The other ones are sources are asynchronous bindings, usually requiring user interactions. These sources are called push sources. They are now limited to the attributes provided at SSO requests when the IdP acts as a SAML2 SP. There attributes are put/found in the Django session.**

Each source in the system is declared with an instance of the AttributeSource model. We'll see later that to forward attributes of push sources it is not necessary that a source is declared in some circumstances.

To manage these sources an attribute policy is attached to services providers. Then the service provider model must be extended with a attribute attributes\_at\_sso\_policy. The service provider must send the signal 'add\_attributes\_to\_response'.

The implementation is actually done for SAML2 providers.

**In such a policy attributes from pull and push sources are treated differently.**

**For pull sources, a list of attributes is indicated. Either an attribute is searched in all the pull sources and whatever attribute value found is returned. Or each attribute is indicated with a source. With each attribute is indicated the output format and namespace.**

**The policy may also indicate that all the attributes in the Django session must be forwarded. Then, no AttributeSource instance is required. All the attributes are then forwarded without treating input namespace considerations. When an AttributeSource instance is found, the input namespace of this source is considered. An option can then be set to tell that the output format and namespace must be taken. A list of attribute can also be given. This list can be use to filter attributes to forward without or without taking care of the source. The output namespace and format can also be trated per attribute.**

If the namespace is default, the attribute names will be taken from the system namespace. In BASIC the name will be the definition name. In URI, the Name will be the OID in urn format and the friendly name will be the definition name. If a namespace is given, the first identifier of this attribute is taken as Name in BASIC. In URI, the same and the first friendly name is taken.

```
class LibertyServiceProvider(models.Model):
    ...
    attribute_policy = models.ForeignKey(AttributePolicy,
        verbose_name=_("Attribute policy"), null=True, blank=True)
```

```

class AttributePolicy(models.Model):
    # List of attributes to provide from pull sources at SSO Login.
    # If an attribute is indicate without a source, from any source.
    # The output format and namespace is given by each attribute.
    attribute_list_for_sso_from_pull_sources = \
        models.ForeignKey(LibertyAttributeMap,
            related_name = "attributes of pull sources",
            blank = True, null = True)

    # Set to true for proxying attributes from pull sources at SSO Login.
    # Attributes are in session.
    # All attributes are forwarded as is except if the parameter
    # 'attribute_list_for_sso_from_push_sources' is initialized
    forward_attributes_from_pull_sources = models.BooleanField(default=False)

    # Map attributes in session
    # forward_attributes_in_session must be true
    # At False, all attributes are forwarded as is
    # At true, look for the namespace of the source for input, If not found,
    # system namespace. Look for the options attribute_name_format and
    # output_namespace of the attribute policy for output.
    map_attributes_from_pull_sources = models.BooleanField(default=False)

    # ATTRIBUTE_VALUE_FORMATS[0] =>
    # (lasso.SAML2_ATTRIBUTE_NAME_FORMAT_BASIC, 'SAMLv2 BASIC')
    output_name_format = models.CharField(max_length = 100,
        choices = ATTRIBUTE_VALUE_FORMATS,
        default = ATTRIBUTE_VALUE_FORMATS[0])

    #ATTRIBUTES_NS[0] => ('Default', 'Default')
    output_namespace = models.CharField(max_length = 100,
        choices = ATTRIBUTES_NS, default = ATTRIBUTES_NS[0])

    # Filter attributes pushed from source.
    source_filter_for_sso_from_push_sources = \
        models.ManyToManyField(AttributeSource,
            related_name = "attributes of pull sources",
            blank = True, null = True)

    # List of attributes to filter from pull sources at SSO Login.
    attribute_filter_for_sso_from_push_sources = \
        models.ForeignKey(LibertyAttributeMap,
            related_name = "attributes of pull sources",
            blank = True, null = True)

    # The sources of attributes of the previous list are considered.
    # May be used conjointly with 'source_filter_for_sso_from_push_sources'
    filter_source_of_filtered_attributes = models.BooleanField(default=False)

    # To map the attributes of forwarded attributes with the default output
    # format and namespace, use 'map_attributes_from_pull_sources'
    # Use the following option to use the output format and namespace
    # indicated for each attribute.
    map_attributes_of_filtered_attributes = models.BooleanField(default=False)

    # Set to true to take in account missing required attributes
    send_error_and_no_attrs_if_missing_required_attrs = \

```

```
models.BooleanField(default=False)

class Meta:
    verbose_name = _('attribute options policy')
    verbose_name_plural = _('attribute options policies')

class AttributeList(models.Model):
    name = models.CharField(max_length = 40, unique = True)
    attributes = models.ManyToManyField(AttributeItem,
        related_name = "attributes of the list",
        blank = True, null = True)

class AttributeItem(models.Model):
    attribute_name = models.CharField(max_length = 100, choices = ATTRIBUTES,
        default = ATTRIBUTES[0])
    # ATTRIBUTE_VALUE_FORMATS[0] =>
    # (lasso.SAML2_ATTRIBUTE_NAME_FORMAT_BASIC, 'SAMLv2 BASIC')
    output_attribute_name_format = models.CharField(max_length = 100,
        choices = ATTRIBUTE_VALUE_FORMATS,
        default = ATTRIBUTE_VALUE_FORMATS[0])
    #ATTRIBUTES_NS[0] => ('Default', 'Default')
    output_namespace = models.CharField(max_length = 100,
        choices = ATTRIBUTES_NS, default = ATTRIBUTES_NS[0])
    required = models.BooleanField(default=False)
    source = models.ForeignKey(AttributeSource, blank = True, null = True)
```

A list of attributes can also be taken from the service provider metadata and added to 'attribute\_list\_for\_sso\_from\_pull\_sources'. The namespace may be extracted from the metadata. This namespace is then used to look for the corresponding definition and then to provide the attribute in the right namespace. Read attributes from metadata is not yet supported.

For the attributes of pull sources, once the list of attributes is defined, They are loaded in the user profile.

As explained before the attribute\_aggregator loading function send signals to grab dictionary of attributes. Up to know, only the ldap loading function are connected to these signals. The namespace of LDAP sources is assumed to be the same as the system namespace. There is here then no mapping needed. Other kind of sources than LDAP can be defined in attribute aggregator.

To grab attributes from a LDAP the user dn in the LDAP or at least a local identifier in the LDAP is required. For this purpose, each user has alias associated with LDAP source. These aliases must their DN in the LDAP. When the authentication LDAP backend will be taken in account, the dn will be taken directly from the user Model instance.

Each LDAP sources are declared with the binding parameters. The LDAP namespace is always 'Default'.

If an attribute to load is not found and is required the answer should report an error (Not yet implemented).

Attributes in response can also be provided with other means than from an LDAP source. Attributes can be put in the user Django session and then loaded in the profile. An option of the service provier indicate if attributes in the session must be provided to the service provider.

To have the attribute loaded from the session, they must be provided in the session as follows: request.session['attributes'][source\_name] = list()

The source\_name must be the name of an existing instance of an 'AttributeSource'. Such an instance contains a field namespace indicating the namespace of attributes.

This is currently implemented only for the SAML2 service provider module of authentic2. Authsaml2, the SP module, parse the assertion and put the attributes in the session.



```
if not 'multisource_attributes' in request.session:
    request.session['attributes'] = dict{}
request.session['multisource_attributes'] \
    [login.assertion.issuer.content] = attributes
```

Then, Authentic2 can be used as a SAML2 proxy forwarding attributes in assertion, eventually doing a namespace mapping. For this, the option forward attributes in session must be set (by default False).



# **COPYRIGHT**

Authentic and Authentic2 are copyrighted by Entr’ouvert and are licensed through the GNU General Public Licence, version 2 or later. A copy of the whole license text is available in the COPYING file.

The OpenID IdP originates in the project django\_openid\_provider by Roman Barczyński, which is under the Apache 2.0 licence. This imply that you must distribute authentic2 under the AGPL3 licence when distributing this part of the project which is the only AGPL licence version compatible with the Apache 2.0 licence.