uniAuth

Giuseppe De Marco

Jan 30, 2023

AUTHOR'S NOTES:

1 Why I decided to develop this IDP 3

2 General Description 5

3 Implementation specific Features 7

4 Requirements and enviroment 9

5 Example project 11

6 Install uniAuth as a Django app 13

7 Configure the software 15

8 Create Database 17

9 LDAP connection 19

10 Create your own SAML certificates 21

11 Create schemas and superuser 23

12 Run debug server 25

13 Production Environment 27

14 MetadataStore definitions 29

15 Service Providers Federation 31

16 Attribute releases 33

17 Entity Categories 35

18 Name ID Format 37

19 Customize uniAuth 39

20 Localization i18n 41

21 MDQuery 43

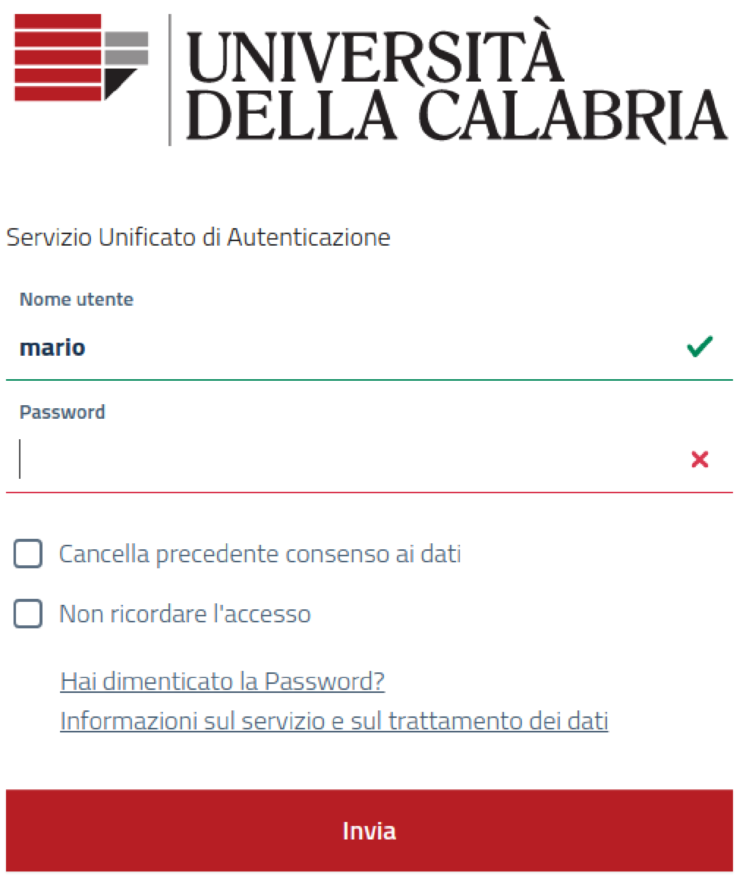
22 AACli 45

23 Backup 47

24 Restore 49

25 Migrate from Shibboleth IdP 51

26 Indices and tables 53



UNIVERSITÀ DELLA CALABRIA

Servizio Unificato di Autenticazione

Nome utente

mario ✓

Password

✗

☐ Cancella precedente consenso ai dati

☐ Non ricordare l'accesso

Hai dimenticato la Password?
Informazioni sul servizio e sul trattamento dei dati
Invia

Github official page is at https://github.com/UniversitaDellaCalabria/uniAuth

CHAPTER ONE

WHY I DECIDED TO DEVELOP THIS IDP

Many SAML2 IDP OpenSource softwares come as mature, I used them and also appreciate them. As a long date Python Programmer I was also looking for something more smart for my needs, at the same time it should have been also very compliant to the standards. For these reasons I choosed to start development on top of Django Framework and pySAML2.

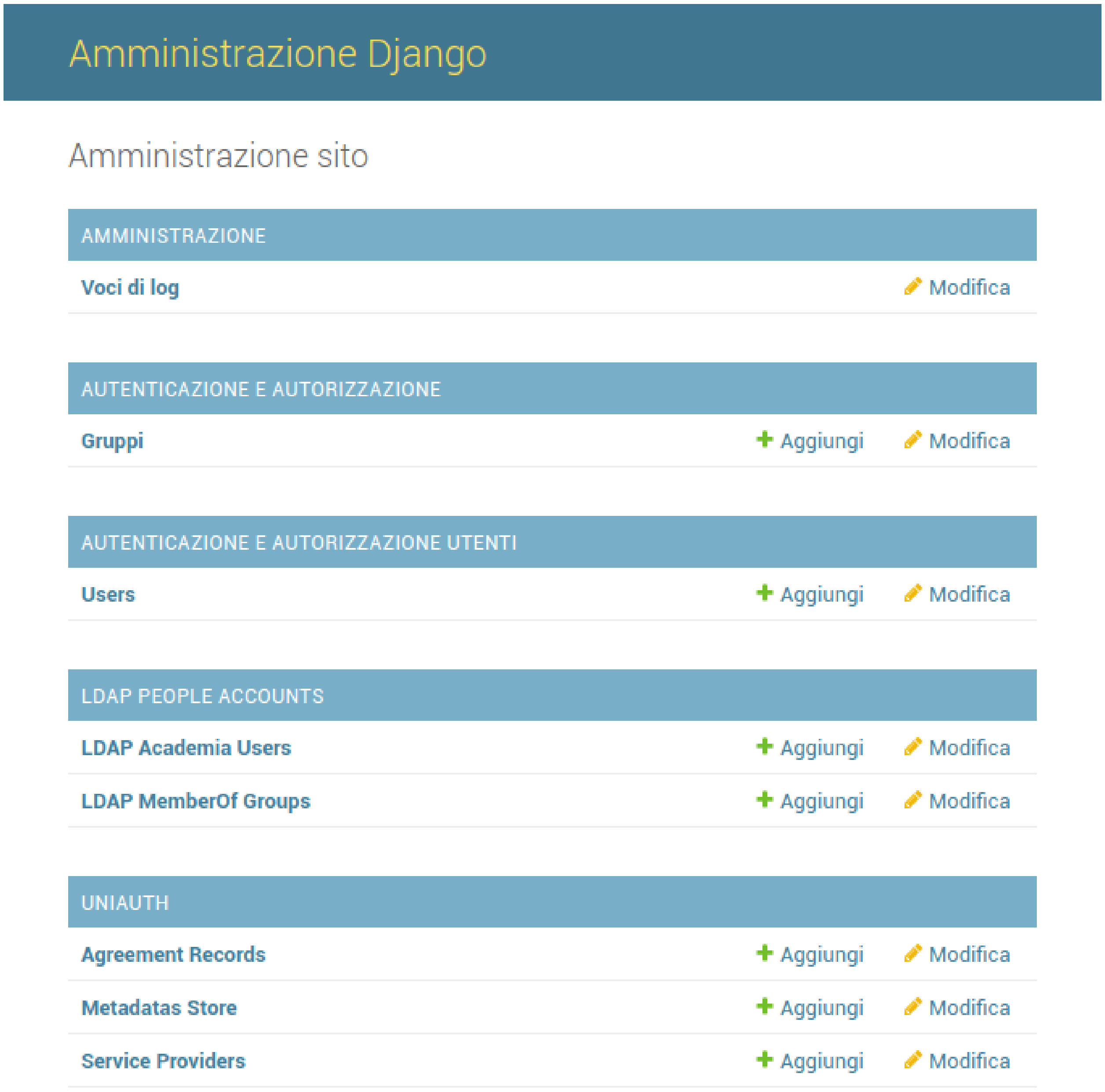
I also noticied that there come always the need to have high sysadmin skills to work with SAML integrated systems, data definitions still need to be stored and handled in multiple files and in a way that, I think, there's too much management costs in time, with repetitive and boring actions handled via console. In addition to this the learning curve related to SAML2 implementations proves itself very slow, often many users preferred to get out of all this.

I also found a lot of python projects developed from scratch and I thought that a Django implementation of them would be a better solution. I decided then to develop an application that would let simple users to do an applicative administration of the platform, create new metadata store and federate new Service Provider, without handle high sysadmin tasks.

Why these great softwares still doesn't have a human management UI and other helpers tools was therefore another of my important questions.

I made contributions in djangosaml2idp. Soon those contributions became a distinct fork, so uniAuth was born as a djangosaml2idp fork because that project won't need some of the features that we found today in uniAuth, of which I also needed within a reasonable time.

My attempt with uniAuth was that to bring the IDentity management to smart users without give up smartness, in the innovation of ordinary management processes. Probably you noticed that uniAuth not come as a Django app but as an entire project, this is because we want to offer a ready-to-use software and not a software too much linked to programming skills of users.



Amministrazione Django

Amministrazione sito

AMMINISTRAZIONE		
Voci di log		Modifica

AUTENTICAZIONE E AUTORIZZAZIONE		
Gruppi	+ Aggiungi	Modifica

AUTENTICAZIONE E AUTORIZZAZIONE UTENTI		
Users	+ Aggiungi	Modifica

LDAP PEOPLE ACCOUNTS		
LDAP Academia Users	+ Aggiungi	Modifica
LDAP MemberOf Groups	+ Aggiungi	Modifica

UNIAUTH		
Agreement Records	+ Aggiungi	Modifica
Metadatas Store	+ Aggiungi	Modifica
Service Providers	+ Aggiungi	Modifica

Fig. 1: Admin backend preview, a daily IDP administration will give you everything you need without touching the console.

CHAPTER

TWO

GENERAL DESCRIPTION

uniAuth, as a SAML2 IDP, is based on pysaml2 and it supports:

- HTTP-REDIRECT and POST bindings (signed authn request must be in HTTP-POST binding);
- ForceAuthn;
- SLO, SAML Single Logout;
- Signed and Encrypted assertions in Response;
- AllowCreate, nameid is stored if nameid format is persistent.

CHAPTER

THREE

IMPLEMENTATION SPECIFIC FEATURES

- no restart is needed when add a new metadata or Service Provider Definition;
- Full Internazionalization support (i18n);
- Interactive Metadata Store definitions through the Admin Backend UI;
- Interactive ServiceProvider definition through the Admin Backend UI;
- Customizable Template and style based on [AGID guidelines](https://www.agid.gov.it/it/argomenti/linee-guida-design-pa);
- MetadataStore and SP validations on save, to prevent faulty configurations in production environment;
- Configurable digest algorithm and salt for Computed NameID;
- **Many configurable options, for every SP we can decide:**
  - enable/disable explicitally;
  - signature and digest algorithms;
  - attributes release (force a set or release what requested by sp);
  - attribute rewrite and creation, fully configurable AttributeProcessors per SP, every aspect of attribute release can be customized from scratch;
  - agreement screen message, availability, data consent form.
- Configurable log rotation through uwsgi;
- Importable StoredPersistentID for each user, from migrations from another IDP;
- An optional LDAP web manager with a configurable app (*ldap_peoples*) through django-ldap-academia-ou-manager;
- Multiple LDAP sources through pyMultiLDAP;
- Detailed logs.

CHAPTER FOUR

REQUIREMENTS AND ENVIROMENT

Install madiadb or whatever RDBMS supported by django ORM

```
sudo apt install xmlsec1 mariadb-server libmariadbclient-dev python3-dev python3-pip␣
↪libssl-dev libmariadb-dev-compat libsasl2-dev libldap2-dev

pip3 install virtualenv
virtualenv -ppython3 uniauth.env
source uniauth.env/bin/activate
```

CHAPTER

FIVE

EXAMPLE PROJECT

```
git clone https://github.com/UniversitaDellaCalabria/uniAuth.git
cd uniAuth
pip3 install -r requirements.txt
pip3 install -r requirements-customizations.txt
cd example/
./manage.py migrate
./manage.py createsuperuser
./manage.py runserver
```

CHAPTER

SIX

INSTALL UNIAUTH AS A DJANGO APP

```
pip install uniauth-saml2-idp
```

CHAPTER

SEVEN

CONFIGURE THE SOFTWARE

You have to copy and edit the following files to have your configuration. The Database and all the Django settings ca be managed in *settingslocal.py*. SAML2 IdP and AA configuration must be configured in idp_pysaml2.py

```
cd django_idp

# copy and modify as your needs
cp settingslocal.py.example settingslocal.py

# copy and modify SAML2 IDP paramenters
cp idp_pysaml2.py.example idp_pysaml2.py
```

djangosaml2 parameters:

SAML_IDP_CONFIG = {}
: the PySAML2 IdP configuration, see *example/django_idp/idp_pysaml2.py.example* and pysaml2 official documentation.

SAML_IDP_DJANGO_USERNAME_FIELD = 'username'
: Attribute used for SAML nameid. It must be a field name, a @property or a callable of the Django User model.

SAML_COMPUTEDID_HASHALG = 'sha256'
: Global behaviour, which algorithm should be used to produce the computedID of a user. Used only for OPAQUE, TRANSIENT and PERSISTENT nameid format.

SAML_COMPUTEDID_SALT = b'87sdf+ybDS+FDSFsdf__7yb'
: Salt used to produce the computed id. Use `b''` to disable salt. Used only for TRANSIENT and PERSISTENT nameid format.

SAML_ALLOWCREATE = True
: If enabled and nameid format is persistent the nameid related to user:recipient_id will be stored in PersistentId model

Platform specific parameters, each of these can be overriden in ServiceProvider configurations:

SAML_IDP_SHOW_USER_AGREEMENT_SCREEN = True
: Global behaviour, show or not the agreement screen.

SAML_IDP_SHOW_CONSENT_FORM = False
: Global behaviour, show or not the form for the consent to transmit the attributes.

SAML_IDP_USER_AGREEMENT_ATTR_EXCLUDE = []
: Global behaviour, if for some reason some attribute should be hidden in the agreement screen (discouraged!).

SAML_IDP_USER_AGREEMENT_VALID_FOR = 24 * 365
: User agreements will be valid for 1 year unless overriden. If this attribute is not used, user agreements will not expire.

SAML_AUTHN_SIGN_ALG and SAML_AUTHN_DIGEST_ALG
Global behaviour, which algorithms should be used for SAML signature and digest.

SAML_FORCE_ENCRYPTED_ASSERTION = False
It will only release encryoted assertion, default = False. SP without encryption key will not works with this configuration.

SAML_DISALLOW_UNDEFINED_SP = True
Only configured SP are allowed to do Authentication requests. If `False` all the SP available in the MetadataStore can request an authentication.

DEFAULT_SPCONFIG = {
Default configuration that will be preloaded on every ServiceProvider configurations. Put here your favourite Attribute Processor or choose another one, from one of your custom application. See examples.

To configure new Metadata stores and federate new Service Providers you can use metadata and SP definitions in `idp_pysaml2.py` for pysaml2 compatibility, otherwise you can create and manage them via Django Admin backend. See dedicated sections for examples.

CREATE DATABASE

You can even use sqlite3 for test purpose. If you want to use mariadb instead, create first the database and the user with the grants, then carry these parameters in your *settingslocal.py* file.

```
# create your MysqlDB
export USER='that-user'
export PASS='that-password'
export HOST='%'
export DB='uniauth'

# tested on Debian 10
sudo mysql -u root -e "\
CREATE USER IF NOT EXISTS '${USER}'@'${HOST}' IDENTIFIED BY '${PASS}';\
CREATE DATABASE IF NOT EXISTS ${DB} CHARACTER SET = 'utf8' COLLATE = 'utf8_general_ci';\
GRANT ALL PRIVILEGES ON ${DB}.* TO '${USER}'@'${HOST}';"
```

CHAPTER NINE

LDAP CONNECTION

You can use LDAP data source using `ldap_peoples` ldap manager or `pyMultiLDAP` apps. If you don't need a LDAP data source remove `ldap_peoples` or `multildap` from `settingslocal.INSTALLED_APPS`.

`ldap_peoples` is a fancy app to integrate a R&S LDAP manager. On top of it you'll find a custom authentication backend and a custom attribute processor, you can even write your custom auth backend and processor with your preferred LDAP library. If you need a fully compliant LDAP configuration with `ldap_peoples` please try the dedicated playbook for it.

If you need multiple LDAP data sources following `ldap_peoples` approach you'll have to create your own django application and use types and methods found in `ldap_peoples`.

If you do not want to create other django application or develop other things to manage multiple LDAP sources, you can use pyMultiLDAP as a proxy, through slapd-sock, or as a python LDAP Client. See *settingslocal.py.example* to have some usage examples.

CHAPTER TEN

CREATE YOUR OWN SAML CERTIFICATES

Then copy them to `certificates` folder and define them in idp_pysaml2.py (`key_file` and `cert_file`, even in `encryption_keypairs`).

```
openssl req -nodes -new -x509 -newkey rsa:2048 -days 3650 -keyout private.key -out␣
↪public.cert
```

CHAPTER

ELEVEN

CREATE SCHEMAS AND SUPERUSER

```
./manage.py migrate
./manage.py createsuperuser
```

CHAPTER TWELVE

RUN DEBUG SERVER

```
./manage.py runserver
```

. . . need a SP for a preliminar tests? see djangosaml2_sp here: https://github.com/peppelinux/Django-Identity

Admin ui could be configured in *settingslocal.py*, with the variable *ADMIN_PATH*. If it is not defined, default will be *admin/*.

CHAPTER THIRTEEN

PRODUCTION ENVIRONMENT

See *uwsgi_setup* examples.

Remember to run `collectstatic` to copy all the static files in the production static folder:

```
./manage.py collectstatic
```

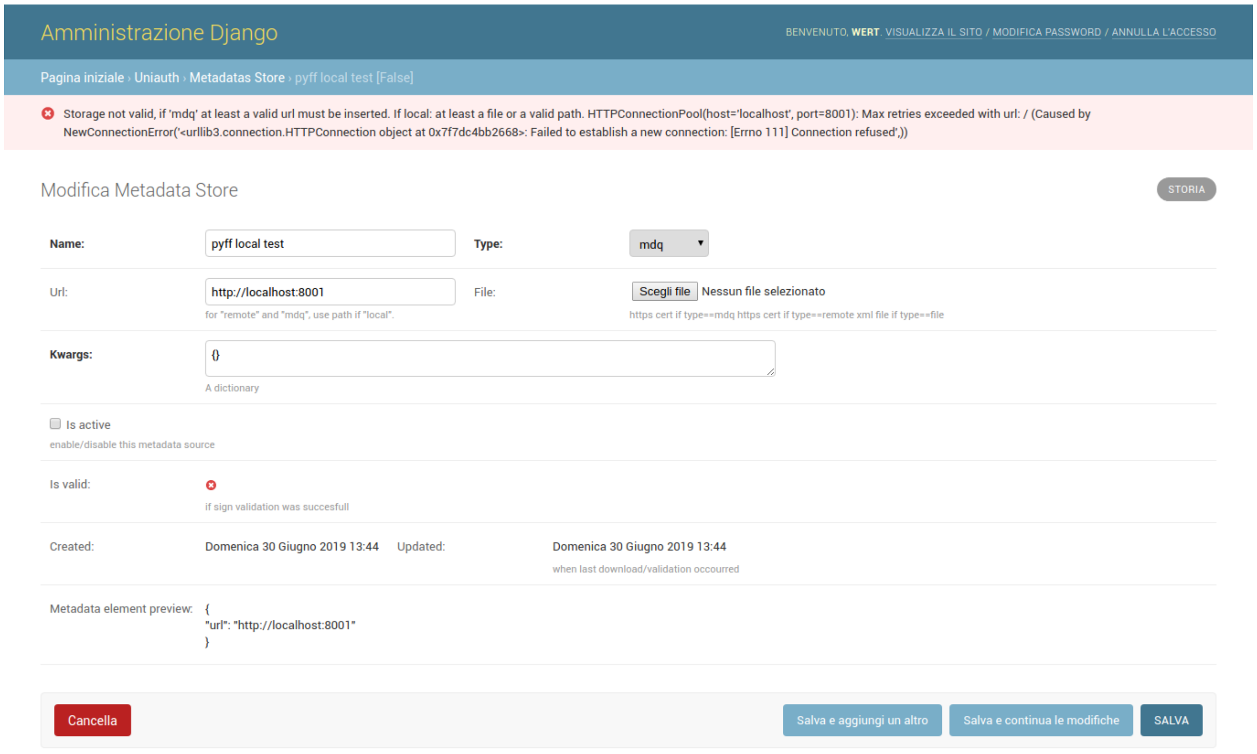
If you need more debug control with the same production configuration, using uwsgi you could run the following commands (absolute paths as examples):

```
/etc/init.d/unicalauth stop
uwsgi --ini /opt/unicalauth/uwsgi_setup/uwsgi.ini.debug
```

CHAPTER

FOURTEEN

METADATASTORE DEFINITIONS



Amministrazione Django

BENVENUTO, **WERT**. VISUALIZZA IL SITO / MODIFICA PASSWORD / ANNULLA L'ACCESSO

Pagina iniziale › Uniauth › Metadatas Store › pyff local test [False]

Storage not valid, if 'mdq' at least a valid url must be inserted. If local: at least a file or a valid path. HTTPConnectionPool(host='localhost', port=8001): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f7dc4bb2668>: Failed to establish a new connection: [Errno 111] Connection refused',))

Modifica Metadata Store

STORIA

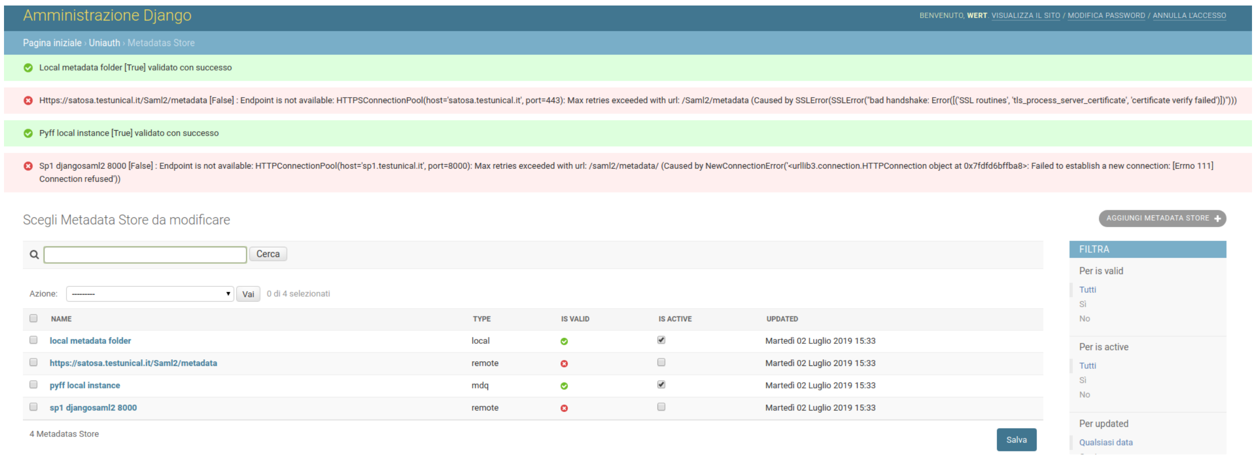
Name:	pyff local test	**Type:**	mdq
Url:	http://localhost:8001 for "remote" and "mdq", use path if "local".	File:	Scegli file Nessun file selezionato https cert if type==mdq https cert if type==remote xml file if type==file
Kwargs:	{} A dictionary		

Is active
enable/disable this metadata source

Is valid:	✗ if sign validation was succesfull		
Created:	Domenica 30 Giugno 2019 13:44	Updated:	Domenica 30 Giugno 2019 13:44 when last download/validation occourred

Metadata element preview:
```
{
"url": "http://localhost:8001"
}
```

Cancella · Salva e aggiungi un altro · Salva e continua le modifiche · SALVA



Amministrazione Django

BENVENUTO, **WERT**. VISUALIZZA IL SITO / MODIFICA PASSWORD / ANNULLA L'ACCESSO

Pagina iniziale › Uniauth › Metadatas Store

Local metadata folder [True] validato con successo

Https://satosa.testunical.it/Saml2/metadata [False] : Endpoint is not available: HTTPSConnectionPool(host='satosa.testunical.it', port=443): Max retries exceeded with url: /Saml2/metadata (Caused by SSLError(SSLError("bad handshake: Error([('SSL routines', 'tls_process_server_certificate', 'certificate verify failed')])")))

Pyff local instance [True] validato con successo

Sp1 djangosaml2 8000 [False] : Endpoint is not available: HTTPConnectionPool(host='sp1.testunical.it', port=8000): Max retries exceeded with url: /saml2/metadata/ (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7fdfd6bffba8>: Failed to establish a new connection: [Errno 111] Connection refused'))

Scegli Metadata Store da modificare

AGGIUNGI METADATA STORE

Cerca

Azione: --------- Vai 0 di 4 selezionati

NAME	TYPE	IS VALID	IS ACTIVE	UPDATED
local metadata folder	local	✓	✓	Martedì 02 Luglio 2019 15:33
https://satosa.testunical.it/Saml2/metadata	remote	✗		Martedì 02 Luglio 2019 15:33
pyff local instance	mdq	✓	✓	Martedì 02 Luglio 2019 15:33
sp1 djangosaml2 8000	remote	✗		Martedì 02 Luglio 2019 15:33

4 Metadatas Store

Salva

FILTRA

Per is valid
Tutti
Sì
No

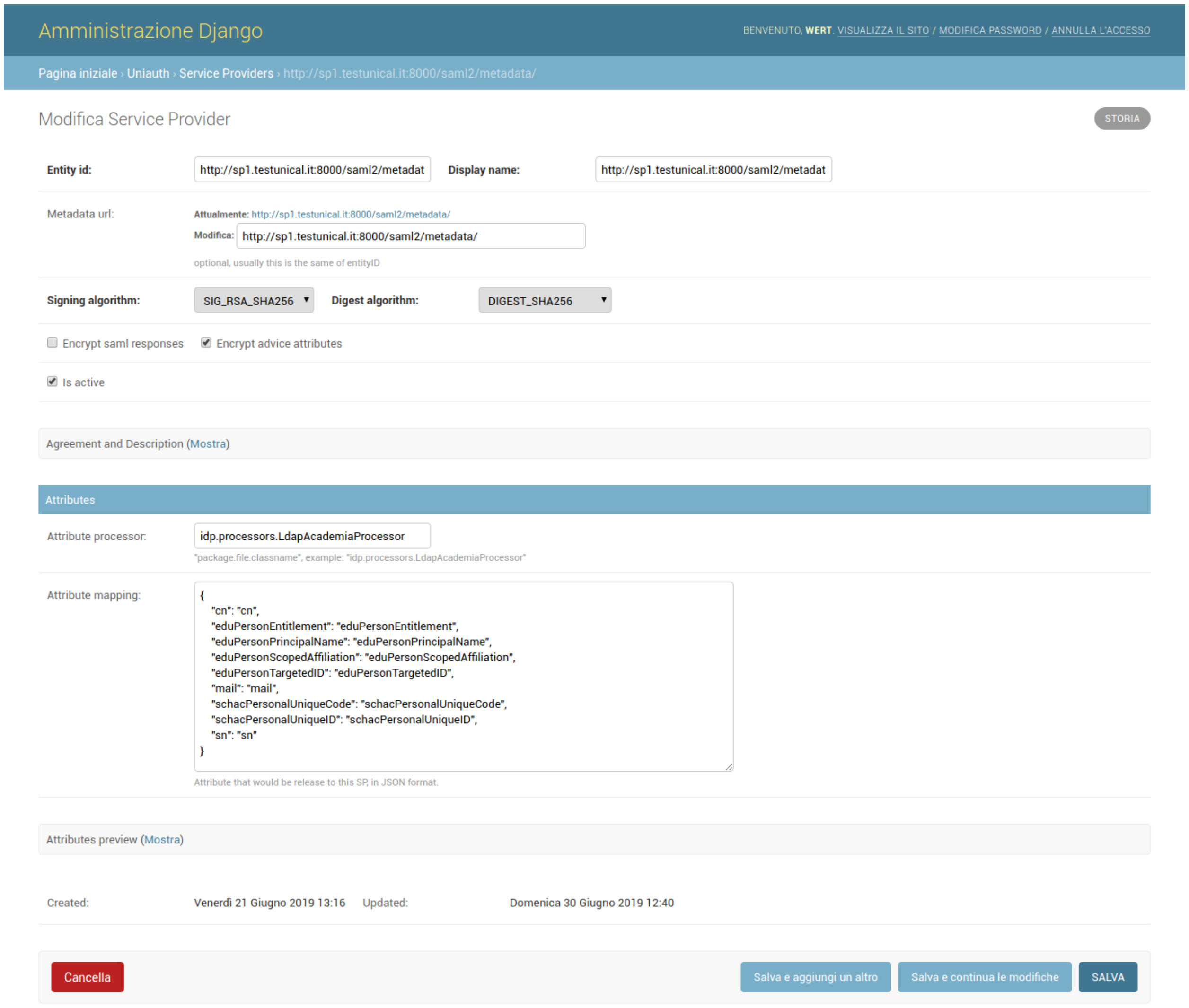
Per is active
Tutti
Sì
No

Per updated
Qualsiasi data

CHAPTER

FIFTEEN

SERVICE PROVIDERS FEDERATION


Amministrazione Django
BENVENUTO, WERT. VISUALIZZA IL SITO / MODIFICA PASSWORD / ANNULLA L'ACCESSO
Pagina iniziale › Uniauth › Service Providers › http://sp1.testunical.it:8000/saml2/metadata/
STORIA
Cancella
Salva e aggiungi un altro
Salva e continua le modifiche
SALVA

Modifica Service Provider

Entity id: http://sp1.testunical.it:8000/saml2/metadat Display name: http://sp1.testunical.it:8000/saml2/metadat

Metadata url: Attualmente: http://sp1.testunical.it:8000/saml2/metadata/
Modifica: http://sp1.testunical.it:8000/saml2/metadata/
optional, usually this is the same of entityID

Signing algorithm: SIG_RSA_SHA256 Digest algorithm: DIGEST_SHA256

☐ Encrypt saml responses ☑ Encrypt advice attributes

☑ Is active

Agreement and Description (Mostra)

Attributes

Attribute processor: idp.processors.LdapAcademiaProcessor
"package.file.classname", example: "idp.processors.LdapAcademiaProcessor"

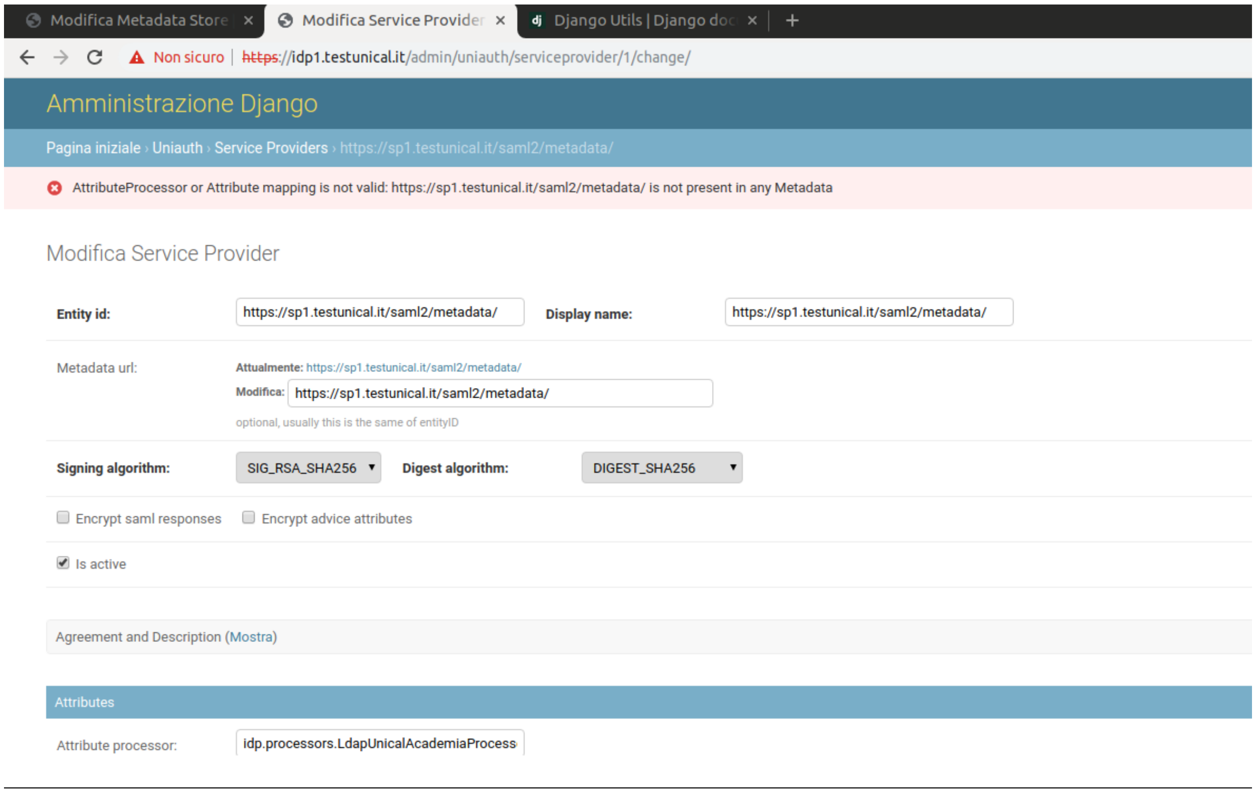
Attribute mapping:

```
{
    "cn": "cn",
    "eduPersonEntitlement": "eduPersonEntitlement",
    "eduPersonPrincipalName": "eduPersonPrincipalName",
    "eduPersonScopedAffiliation": "eduPersonScopedAffiliation",
    "eduPersonTargetedID": "eduPersonTargetedID",
    "mail": "mail",
    "schacPersonalUniqueCode": "schacPersonalUniqueCode",
    "schacPersonalUniqueID": "schacPersonalUniqueID",
    "sn": "sn"
}
```

Attribute that would be release to this SP, in JSON format.

Attributes preview (Mostra)

Created: Venerdì 21 Giugno 2019 13:16 Updated: Domenica 30 Giugno 2019 12:40



Modifica Metadata Store | Modifica Service Provider | Django Utils | Django doc

Non sicuro | ~~https~~://idp1.testunical.it/admin/uniauth/serviceprovider/1/change/

Amministrazione Django

Pagina iniziale › Uniauth › Service Providers › https://sp1.testunical.it/saml2/metadata/

AttributeProcessor or Attribute mapping is not valid: https://sp1.testunical.it/saml2/metadata/ is not present in any Metadata

Modifica Service Provider

Entity id:	https://sp1.testunical.it/saml2/metadata/	**Display name:**	https://sp1.testunical.it/saml2/metadata/
Metadata url:	**Attualmente:** https://sp1.testunical.it/saml2/metadata/ **Modifica:** https://sp1.testunical.it/saml2/metadata/ optional, usually this is the same of entityID		
Signing algorithm:	SIG_RSA_SHA256	**Digest algorithm:**	DIGEST_SHA256

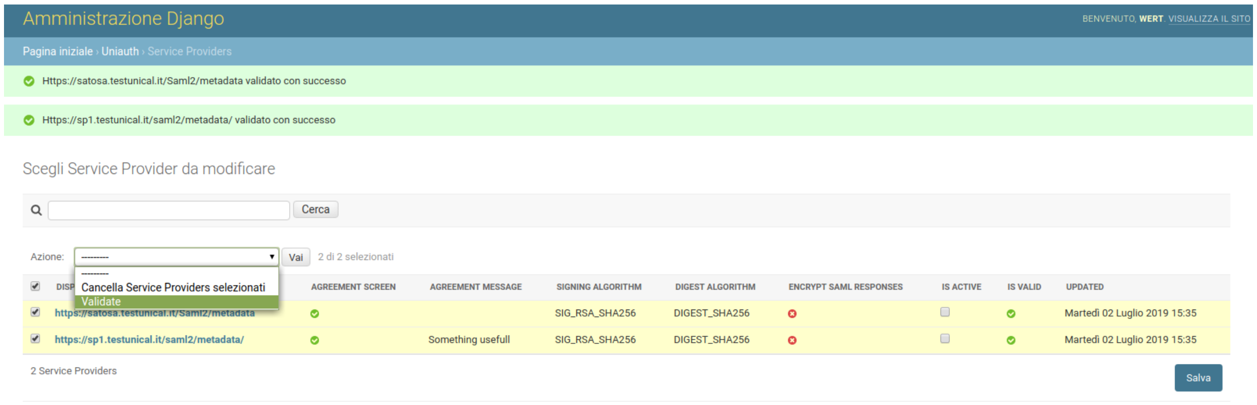
☐ Encrypt saml responses ☐ Encrypt advice attributes

☑ Is active

Agreement and Description (Mostra)

Attributes

Attribute processor: idp.processors.LdapUnicalAcademiaProcess



Amministrazione Django

BENVENUTO, **WERT**. VISUALIZZA IL SITO

Pagina iniziale › Uniauth › Service Providers

Https://satosa.testunical.it/Saml2/metadata validato con successo

Https://sp1.testunical.it/saml2/metadata/ validato con successo

Scegli Service Provider da modificare

Azione: --------- Vai 2 di 2 selezionati

☑	DISP	AGREEMENT SCREEN	AGREEMENT MESSAGE	SIGNING ALGORITHM	DIGEST ALGORITHM	ENCRYPT SAML RESPONSES	IS ACTIVE	IS VALID	UPDATED
☑	https://satosa.testunical.it/Saml2/metadata	✓		SIG_RSA_SHA256	DIGEST_SHA256	✗	☐	✓	Martedì 02 Luglio 2019 15:35
☑	https://sp1.testunical.it/saml2/metadata/	✓	Something usefull	SIG_RSA_SHA256	DIGEST_SHA256	✗	☐	✓	Martedì 02 Luglio 2019 15:35

2 Service Providers

CHAPTER

SIXTEEN

ATTRIBUTE RELEASES

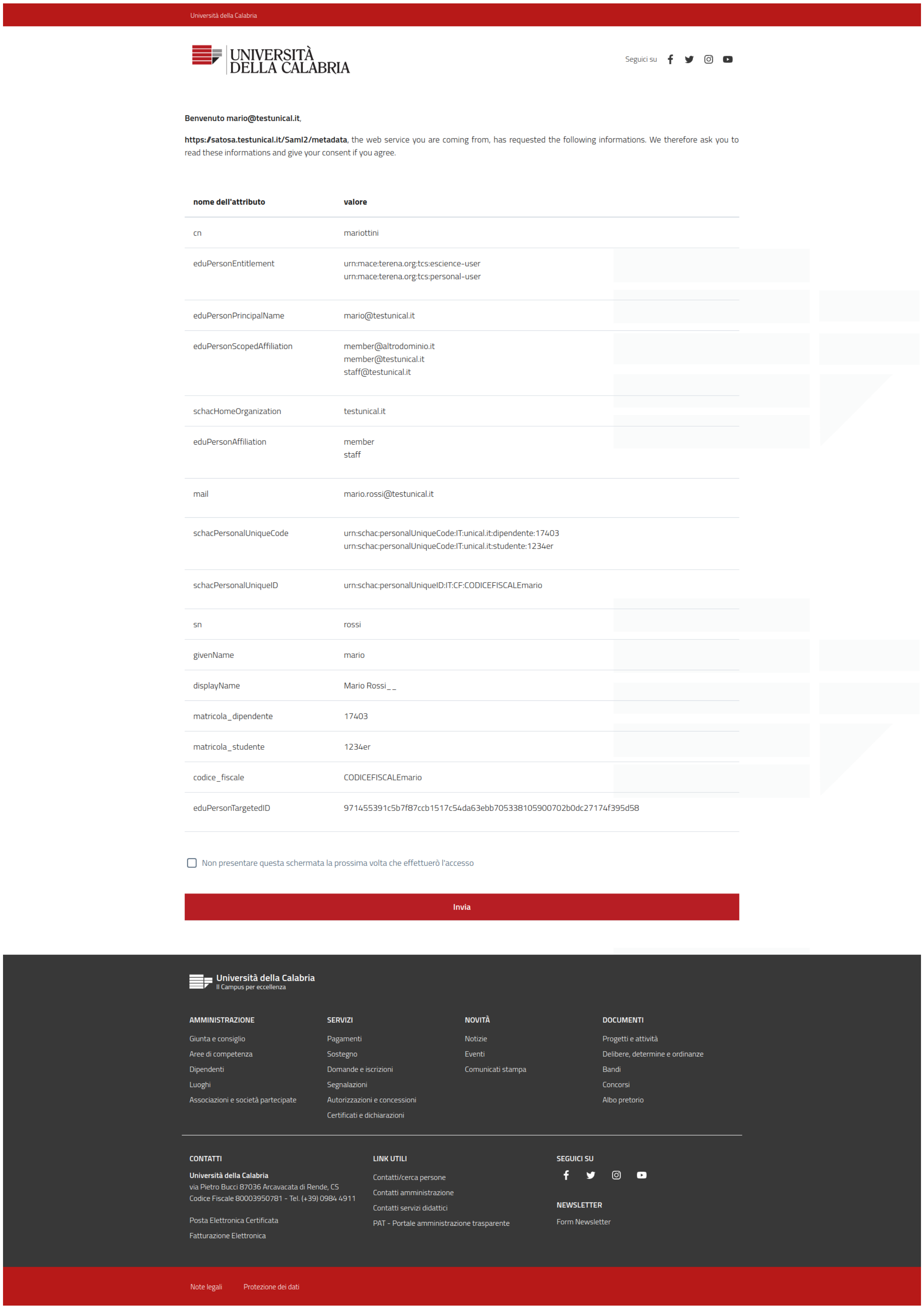
By default IdP will only release required Attributes defined in each SP metadata (isRequired=True or EntityCategories), if they are available. Otherwise the IdP will release a default attribute set, defined in settings parameters. It can also force some attribute release by checking `force_attribute_release` into each SP configuration.

Every SP can use a specific Attribute Processor, you can even customize a brand new one in an application that can be easily installed into `django_idp.settingslocal.INSTALLED_APPS`. You can see how these `processors` works simply looking at `uniauth_saml2_idp.base.processors` and `uniauth_saml2_idp.ldap.processors`.

The Attribute Processor can fetch data from third-party sources and manipulate attributes as well.

There also a special class named `NameIdBuilder`, the nameID policy relies on it, it should be very easy to inherit and customize as needed.

In every `processors` there's a special method called `extra_attr_processing` where to put additional conditions and values processing. See `idp.processors.LdapUnicalAcademiaProcessor` for an example of inheritance with the use of this method.



Università della Calabria

UNIVERSITÀ DELLA CALABRIA

Seguici su

Benvenuto mario@testunical.it,

https://satosa.testunical.it/Saml2/metadata, the web service you are coming from, has requested the following informations. We therefore ask you to read these informations and give your consent if you agree.

nome dell'attributo	valore
cn	mariottini
eduPersonEntitlement	urn:mace:terena.org:tcs:escience-user urn:mace:terena.org:tcs:personal-user
eduPersonPrincipalName	mario@testunical.it
eduPersonScopedAffiliation	member@altrodominio.it member@testunical.it staff@testunical.it
schacHomeOrganization	testunical.it
eduPersonAffiliation	member staff
mail	mario.rossi@testunical.it
schacPersonalUniqueCode	urn:schac:personalUniqueCode:IT:unical.it:dipendente:17403 urn:schac:personalUniqueCode:IT:unical.it:studente:1234er
schacPersonalUniqueID	urn:schac:personalUniqueID:IT:CF:CODICEFISCALEmario
sn	rossi
givenName	mario
displayName	Mario Rossi_ _
matricola_dipendente	17403
matricola_studente	1234er
codice_fiscale	CODICEFISCALEmario
eduPersonTargetedID	971455391c5b7f87ccb1517c54da63ebb705338105900702b0dc27174f395d58

☐ Non presentare questa schermata la prossima volta che effettuerò l'accesso

Invia

Università della Calabria
Il Campus per eccellenza

AMMINISTRAZIONE	SERVIZI	NOVITÀ	DOCUMENTI
Giunta e consiglio	Pagamenti	Notizie	Progetti e attività
Aree di competenza	Sostegno	Eventi	Delibere, determine e ordinanze
Dipendenti	Domande e iscrizioni	Comunicati stampa	Bandi
Luoghi	Segnalazioni		Concorsi
Associazioni e società partecipate	Autorizzazioni e concessioni		Albo pretorio
	Certificati e dichiarazioni		

CONTATTI

Università della Calabria
via Pietro Bucci 87036 Arcavacata di Rende, CS
Codice Fiscale 80003950781 - Tel. (+39) 0984 4911

Posta Elettronica Certificata

Fatturazione Elettronica

LINK UTILI

Contatti/cerca persone

Contatti amministrazione

Contatti servizi didattici

PAT - Portale amministrazione trasparente

SEGUICI SU

NEWSLETTER

Form Newsletter

Note legali   Protezione dei dati

CHAPTER SEVENTEEN

ENTITY CATEGORIES

Entity Categories is handled as it come from pySAML2. In the *django_idp.idp_pysaml2* we can define `entity_category_support` or `entity_category` as follow

```
SAML_IDP_CONFIG = {
    'debug' : True,
    'xmlsec_binary': get_xmlsec_binary(['/opt/local/bin', '/usr/bin/xmlsec1']),
    'entityid': '%s/metadata' % BASE_URL,
    'attribute_map_dir': 'data/attribute-maps',
    'description': 'SAML2 IDP',

    'entity_category': [edugain.COCO, # "http://www.geant.net/uri/dataprotection-code-of-
↪conduct/v1"
                        refeds.RESEARCH_AND_SCHOLARSHIP],

    'service': {
```

The previous configuration will expose Entity Categories in the IDP metadata. If we need also to handle these as policy, to manage these as restrictions on attribute release, we could define them in `SAML_IDP_CONFIG['service']['idp']['policy']`

```
"policy": {
    "default": {
        "lifetime": {"minutes": 15},
        "name_form": NAME_FORMAT_URI,
        # if the sp are not conform to entity_categories (in our metadata)
        # the attributes will not be released
        # "entity_categories": ["refeds",],
    },

    # attributes will be released only if this SP have
    # edugain entity_category definition in its metadata.
    "https://sp1.testunical.it/saml2/metadata/": {
        "entity_categories": ["edugain"]
    }

}
```

CHAPTER

EIGHTEEN

NAME ID FORMAT

This uniAuth release only supports these Name ID formats:

- NAMEID_FORMAT_UNSPECIFIED
- NAMEID_FORMAT_TRANSIENT
- NAMEID_FORMAT_PERSISTENT
- NAMEID_FORMAT_EMAILADDRESS

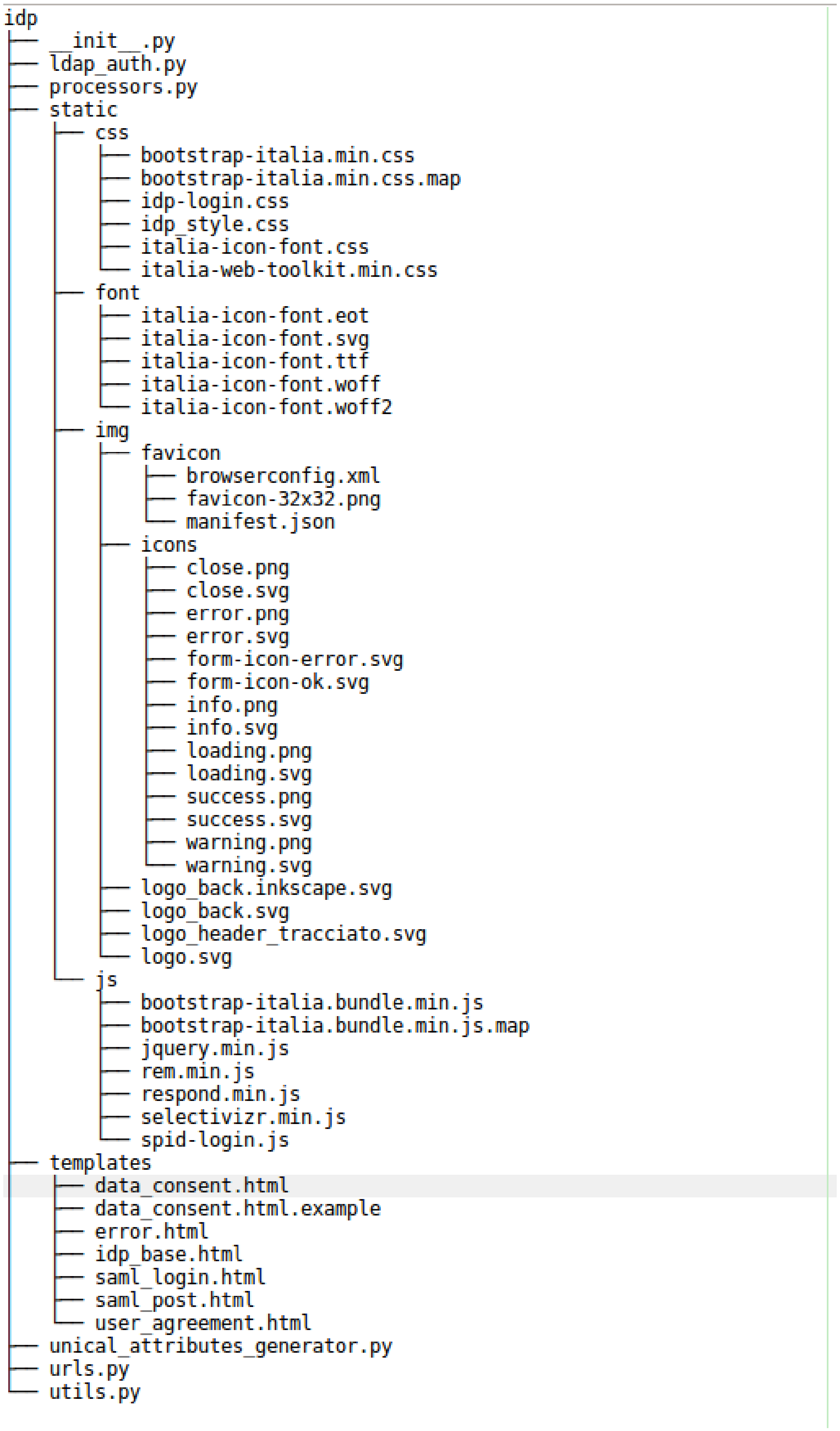
See `uniauth_saml2_idp.base.processors.NameIdBuilder` if you need to implement other formats, it's trivial.

CHAPTER

NINETEEN

CUSTOMIZE UNIAUTH

In the projects tree there's an example project called *example*. It come with an application callend *uniauth_unical_template* in `django_idp.settingslocal.INSTALLED_APPS` where we have all the html template and static files. Start from this example template for doing your customizations.



```
idp
├── __init__.py
├── ldap_auth.py
├── processors.py
├── static
│   ├── css
│   │   ├── bootstrap-italia.min.css
│   │   ├── bootstrap-italia.min.css.map
│   │   ├── idp-login.css
│   │   ├── idp_style.css
│   │   ├── italia-icon-font.css
│   │   └── italia-web-toolkit.min.css
│   ├── font
│   │   ├── italia-icon-font.eot
│   │   ├── italia-icon-font.svg
│   │   ├── italia-icon-font.ttf
│   │   ├── italia-icon-font.woff
│   │   └── italia-icon-font.woff2
│   ├── img
│   │   ├── favicon
│   │   │   ├── browserconfig.xml
│   │   │   ├── favicon-32x32.png
│   │   │   └── manifest.json
│   │   ├── icons
│   │   │   ├── close.png
│   │   │   ├── close.svg
│   │   │   ├── error.png
│   │   │   ├── error.svg
│   │   │   ├── form-icon-error.svg
│   │   │   ├── form-icon-ok.svg
│   │   │   ├── info.png
│   │   │   ├── info.svg
│   │   │   ├── loading.png
│   │   │   ├── loading.svg
│   │   │   ├── success.png
│   │   │   ├── success.svg
│   │   │   ├── warning.png
│   │   │   └── warning.svg
│   │   ├── logo_back.inkscape.svg
│   │   ├── logo_back.svg
│   │   ├── logo_header_tracciato.svg
│   │   └── logo.svg
│   └── js
│       ├── bootstrap-italia.bundle.min.js
│       ├── bootstrap-italia.bundle.min.js.map
│       ├── jquery.min.js
│       ├── rem.min.js
│       ├── respond.min.js
│       ├── selectivizr.min.js
│       └── spid-login.js
├── templates
│   ├── data_consent.html
│   ├── data_consent.html.example
│   ├── error.html
│   ├── idp_base.html
│   ├── saml_login.html
│   ├── saml_post.html
│   └── user_agreement.html
├── unical_attributes_generator.py
├── urls.py
└── utils.py
```

Fig. 1: This is the structure of *idp*

CHAPTER

TWENTY

LOCALIZATION I18N

It relies to Django documentation.

You'll find gettext .po files into `locale/` folder, then you can translate messages before compiling them with:

```
./manage.py compilemessage
```

CHAPTER

TWENTYONE

MDQUERY

This command permit us to check the availability of a saml entity in the IdP metadata store. The option *-f* can specify the output format, if saml2 (default) or json. It will print the entity metadata in the console.

```
./manage.py mdquery -e "http://sp1.testunical.it:8000/saml2/metadata/"
./manage.py mdquery -e "http://sp1.testunical.it:8000/saml2/metadata/" -f json
```

CHAPTER

TWENTYTWO

AACLI

This feature will let us check wich attributes will be released to a specified Service Provider regarding a specified user.

```
./manage.py aacli -u mario -e https://sptest.auth.unical.it/saml2
```

example output:

```
SP Configuration:
{
  "processor": "uniauth_saml2_idp.processors.ldap.LdapUnicalMultiAcademiaProcessor",
  "attribute_mapping": {
    "cn": "cn",
    "codice_fiscale": "codice_fiscale",
    "displayName": "displayName",
    "eduPersonAffiliation": "eduPersonAffiliation",
    "eduPersonEntitlement": "eduPersonEntitlement",
    "eduPersonHomeOrganization": "eduPersonHomeOrganization",
    "eduPersonPrincipalName": "eduPersonPrincipalName",
    "eduPersonScopedAffiliation": "eduPersonScopedAffiliation",
    "eduPersonTargetedID": "eduPersonTargetedID",
    "email": [
      "mail",
      "email"
    ],
    "givenName": [
      "givenName",
      "another_possible_occourrence"
    ],
    "mail": [
      "mail",
      "email"
    ],
    "matricola_dipendente": "matricola_dipendente",
    "matricola_studente": "matricola_studente",
    "schacHomeOrganization": "schacHomeOrganization",
    "schacPersonalUniqueCode": "schacPersonalUniqueCode",
    "schacPersonalUniqueID": "schacPersonalUniqueID",
    "sn": "sn"
  },
  "force_attribute_release": false,
  "display_name": "http://sp1.testunical.it:8000/saml2/metadata/",
  "display_description": "",
```

(continues on next page)

(continued from previous page)

```
  "display_agreement_message": "",
  "signing_algorithm": "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256",
  "digest_algorithm": "http://www.w3.org/2001/04/xmlenc#sha256",
  "disable_encrypted_assertions": true,
  "show_user_agreement_screen": true,
  "display_agreement_consent_form": false
}

TargetedID: 4b7dc8cc66796e63702f7baa73588f772191254801ab9369b7dfa883dbccad58
{
  "cn": [
    "mario rossi"
  ],
  "eduPersonEntitlement": [
    "urn:mace:terena.org:tcs:personal-user",
    "urn:mace:terena.org:tcs:escience-user",
    "urn:mace:dir:entitlement:common-lib-terms"
  ],
  "eduPersonPrincipalName": [
    "mario@testunical.it"
  ],
  "eduPersonScopedAffiliation": [
    "staff@testunical.it",
    "member@testunical.it",
    "member@altrodominio.it"
  ],
  "email": [
    "mario.rossi@testunical.it"
  ],
  "givenName": [
    "mario"
  ],
  "mail": [
    "mario.rossi@testunical.it"
  ],
  "schacHomeOrganization": [
    "testunical.it"
  ],
  "schacPersonalUniqueCode": [
    "urn:schac:personalUniqueCode:it:testunical.it:dipendente:1237403",
    "urn:schac:personalUniqueCode:it:testunical.it:studente:1234er"
  ],
  "schacPersonalUniqueID": [
    "urn:schac:personalUniqueID:it:CF:CODICEFISCALEmario"
  ],
  "sn": [
    "rossi"
  ],
  "codice_fiscale": "CODICEFISCALEmario"
}
```

CHAPTER TWENTYTHREE

BACKUP

We can export all the MetadataStores, the federated ServiceProviders and user's Agreements in JSON format as follow:

```
./manage.py dumpdata uniauth_saml2_idp
# to a file
./manage.py dumpdata uniauth_saml2_idp > /path/to/file.json
```

If we had some users with legacy SAML persistent ID stored in our `USER_MODEL` we can also backup these with the following command:

```
./manage.py dumpdata accounts
```

CHAPTER TWENTYFOUR

RESTORE

To restore these backups just run this:

```
./manage.py loaddata /path/to/file.json
```

CHAPTER

TWENTYFIVE

MIGRATE FROM SHIBBOLETH IDP

Here a brief description of the general steps to do for migrating an existing Shibboleth IdP to uniAuth, carrying the same configuration. We have migrate from Shibboleth IdP 3.4.6 to uniAuth v2.0.0, here the steps we made to achieve this goal:

1. copy SAML2 certificates, from shibboleth idp SAML in *credentials/* to your pysaml2 configuration.
2. Standing on Shibboleth metadata, in *metadata/idp-metadata.xml*, place the same Service Endpoints urls to your project's urls file:

```
if 'uniauth_saml2_idp' in settings.INSTALLED_APPS:
    import uniauth_saml2_idp.urls
    from uniauth_saml2_idp.views import SsoEntryView, LogoutProcessView

    urlpatterns += path('idp/profile/SAML2/<str:binding>/SSO', SsoEntryView.as_
↪view(),
                name="saml_login_binding"),
    urlpatterns += path('idp/profile/SAML2/<str:binding>/SLO', LogoutProcessView.as_
↪view(),
                name="saml_logout_binding"),
    urlpatterns += path('idp/shibboleth/', metadata, name='saml2_idp_metadata'),

    urlpatterns += path(
        'idp/', include((uniauth_saml2_idp.urls, 'uniauth_saml2_idp',))
    ),
```

3. Configure the same entityID in your pysaml2 configuration.
4. Migrate the existing Shibboleth IdP *conf/attribute-filters.xml* (and any other available in *conf/services.xml*) to uniauth SP definitions (ModelAdmin or settings.py).
5. If you use LDAP: Configure PyMultiLDAP rewrite rules and pattern matching, standing on the Attributes defined in *conf/attribute-resolver.xml* (and any other available in `conf/services.xml).
6. Configure your metadata store (ModelAdmin or settings.py). It's suggested to use a MDQ Server for loading large federation xml files, as to be with eduGain.
7. Use uniauth *aacli* and *mdquery* commands to check the availability of Entities and the attribute to be released to them.

CHAPTER

TWENTYSIX

INDICES AND TABLES

- genindex
- modindex
- search