FORMA LMS

Integration Technical Reference

Autore: Davide Anceschi

Ultimo Aggiornamento: 03/06/2019

Stato: Bozza

Versione: 0.0.303

Summary

1 SSO

Forma is able to support Single Sign-On and Single-Login functions with 3rd party systems even through an ad-hoc methods.

Forma validates a user's credential based on a pre-calculated token (hash) that is sent at the time of the passage from the external application to Forma.

The information contained in the token guarantees the validity and authenticity of the authentication request. If the check is positive, the user is transparently authenticated on the Forma system as well

The parameters for token generation are:

- Forma username
- time
- secret code

The secret code is an alphanumeric string whose content is pre-agreed between the applications involved in the interaction

The token creation is done through the following function:

token = hash md5 (username , time, secret code);

The rules to pass the Parameters to the function are the following:

- The time parameter is: yyyymmddhhmmss
- The username used to create the hash must be in lowercase

2 AUTHENTICATION API

This is the authentication procedure for Forma REST API.

The system is based on a pair of keys called: key and secret

The “key” one will be (clearly) exchanged among the systems, while the “secret” will be used to create the concatenation parameter. The call will be executed as described, but in its header must be added the X-Authorization parameter as follows:

X-Authorization: FormaLMS <code>

To obtain the “<code>” value (that must be concatenated after the “FormaLMS” string) it is requested to proceed as follows:

- create sha1 encoding of this concatenation: post values in the list and separated by comma, the comma itself, the secret key (“secret”)
- generate the "<code>" through a base64 encoding of the “key” concatenation and with the “:” element, plus the token previously generated.

EXAMPLE

```
$codice_sha1 = strtolower(sha1(implode(',', $params) . ',' . $secret));
$codice = base64_encode($key . ':' . $codice_sha1);
```

additional header row for post call:

'X-Authorization: FormaLMS '. $codice

3 API

3.1 Introduction

Summary

Area	Name	Description
User	/api/user/checkUsername	
	/api/user/create	
	/api/user/edit	
	/api/user/delete	
Authentication	/api/auth/authenticate	
User course subscription	/api/user/userCourses	
	/api/course/addUserSubscription	
	/api/course/deleteUserSubscription	

PARAMETERS

Name	Type	Description
Userid	String	user userid (username)

EXAMPLE

curl --header "application/x-www-form-urlencoded" --data "idst=12345&userid=lotta&firstname=Name&lastname=cogName&password=S3gr3t0&email=posta.ele@it" http://localhost/forma/api/user/create/

Input

idst=12345&course_id=67890

Output

```
<XMLoutput>
 <success>true</success>
 <idst>67890</idst>
</XMLoutput>
```

```
<?xml version="1.0" encoding="UTF-8"?><XMLoutput><error>User not found</error></XMLoutput>
```

3.2 /api/user/checkUsername

Return a list of users; if “count” parameters is not specified, all records will be returned

PARAMETERS

Name	Type	Description
userid	String	user userid (username)

EXAMPLE

Input

userid=Nomeutente

Output

```
 <success>true</success>
 <idst>12345</idst>
```

3.3 /api/user/create

Create a new user through the API system; returns the idst of the new user.

PARAMETERS

Name	Type	Description
Userid	string	user userid (username)
firstname	string	
lastname	string	
password	string	
email	string	
role	string	admin = administrator

		pubadmin = publicadmin user = user
valid	integer	0: suspended user; 1: active user
orgchart	string	orgchart node: path of the node for adding the user; if null or empty string, the user is added to root node. Accepts both node name or code
force_change	integer	force password change on login (0,1)
_customfields	array	each array's key is the ID of the custom field. For each ID key add the custom value

EXAMPLE

Input

userid=Nameutente&firstname=mario&lastname=rossi&password=password&email=mario.rossi@email.it&role=admin&valid=1

Output

```
<success>true</success>
<idst>12345</idst>
```

Example Array:

```
$user_array = array(
                'userid'       => 'new_user01',
                'firstname'    => 'Jonh',
                'lastname'     => 'Reed',
                'password'     => 'changeme',
                'email'        => 'jr@testemail.com',
                'role'         => "student",
                'orgchart'     => '/firstlevel_node/',
                'valid'        => 1,
                'force_change'  => 1,
                '_customfields' => [
                                5 => 'custom field 1 value' ,
                                6 => 'custom field 2 value'
                                ]
                );
```

3.4 /api/user/edit

This function update a user through the api system.

Parameters

Name	Type	Description
Idst	integer	numeric user id
userid	string	user userid (username)
firstname	string	
lastname	string	
password	string	
email	String	
valid	integer	0 = suspended / 1 = active
_customfields	array	each array's key is the ID of the custom field. For each ID key add the custom value

EXAMPLE

Input

idst=12345&userid=Nameutente&firstname=mario&lastname=rossi&password=password&email=mario.rossi@email.it&valid=1

Output

```
<success>true</success>
```

3.5 /api/user/delete

Delete a user trough the api system

PARAMETERS:

Name	Type	Description
Idst	Integer	

EXAMPLE

Input

idst=12345

Output

```
<success>true</success>
<message>User #12345 has been deleted.</message>
```

3.6 /api/auth/authenticate

Check if valid username and password are provided and returns the authentication token required for single sign on.

PARAMETERS

Name	Type	Description
username	string	
password	string	

EXAMPLE

Input

username=admin&password=password

Output

```
<success>true</success>
<message>You are authenticated.</message>
<token>string</token>
<expire_at>date</expire_at>
```

3.7 /api/user/userCourses

Return the list of courses user is subscribed to

PARAMETERS

Name	Type	Description
idst	Integer	numeric user id

EXAMPLE

Input

idst=12345&course_id=67890

Output

```
<success>true</success>
<element>
  <course_info>
    <course_id>12345</course_id>
    <course_name>string</course_name>
    <course_description>string</course_description>
    <course_link>string</course_link>
    <user_status>string</user_status>
```

```
   </course_info>
  </element>
</XMLoutput>
```

3.8 /api/course/addUserSubscription

Subscribe a user to a course, edition or classroom.

PARAMETERS:

Name	Type	Description	Optional (default)
Idst	integer	numeric user id	
course_id	integer		
user_level	string	administrator instructor mentor tutor student ghost guest	student

EXAMPLE

Input

idst=12345&course_id=67890&user_level=student

Output

```
 <success>true</success>
 <message>User has been subscribed to the course</message>
```

3.9 /api/course/deleteUserSubscription

Delete user subscription form a course, edition or classroom.

PARAMETERS

Name	Type	Description
Idst	integer	numeric user id
course_id	integer	

EXAMPLE

Input

idst=12345&course_id=67890

Output

<success>true</success>
<message>User has been removed from the course</message>

3.10 /api/course/getCertificateByUser

Get user's certificates by username and optional field id course

PARAMETERS

Name	Type	Description
username	string	Username of user
course_id	integer	Id course

EXAMPLE

Input

username=user&course_id=67890

Output

```
<XMLoutput>
      <success>true</success>
      <idst>12605</idst>
      <firstname>string</firstname>
      <lastname>string</lastname>
      <userid>string</userid>
      <certificate_list>
            <element>
                        <course_id>67890</course_id>
                        <course_code>string</course_code>
                        <course_name>string</course_name>
                        <date_generate>date</date_generate>
                        <cert_file>string</cert_file>
                  </element>


        </certificate_list>
```

```
</XMLoutput>
```

3.11 /api/course/getCertificateByCourse

Get course's certificates by id course and optional field username

PARAMETERS

Name	Type	Description
username	string	Username of user
course_id	integer	Id course

EXAMPLE

Input

username=user&course_id=67890

Output

```
<XMLoutput>
      <success>true</success>
      <course_id>67890</course_id>
      <course_code>string</course_code>
      <course_name>string</course_name>
      <box_description>text</box_description>
      <certificate_list>
            <element>
                        <id_idst>12605</id_course>
                        <fistname>string</fistname>
                        <lastname>string</lastname>
                        <userid>string</userid>
                        <date_generate>date</date_generate>
                        <cert_file>string</cert_file>
                </element>


        </certificate_list>
        </XMLoutput>
```