Use the Ubisecure SSO Management API with curl

This page illustrates the use of the SSO Management API using the curl command line tool. Note that the commands have been run on windows, escaping of special characters and referencing of environment variables will differ depending on the command shell used.

This is an illustrative example only - typically these calls would be made by other scripting languages or orchestration techniques.

Refer also to the API description in the documentation available at SSO Management API.

Step-by-step guide

The first step when using the SSO Management API is to discover the scope required to request in the access token:

```
curl https://sso.example.com/sso-api/site
```

The response is unauthorized and includes the reason why - you must present an access_token with the scope "openid 45625ffa-912d-427d-bf27-2f36fe1c4ecf"

```
HTTP/1.1 401 Unauthorized
Server: Apache-Coyote/1.1
Cache-Control: no-cache, must-revalidate, no-transform, max-age=0, private
WWW-Authenticate: Bearer realm="45625ffa-912d-427d-bf27-2f36fe1c4ecf" scope="openid 45625ffa-912d-427d-bf27-
2f36fe1c4ecf"
Content-Length: 0
Date: Thu, 15 Sep 2016 10:02:04 GMT
```

Get the token using the scope shown above.

The access token can be requested using any supported grant_types. Authorization code flow is recommended and enables single sign-on using the operator credentials.

For the purposes of this example only, we will use the password grant. The client_id and secret of the SSO Management API OAuth2 agent are given in the HTTP Basic user credentials.

```
curl --basic -u 99b5d85b-0ab7-4887-bd70-8e0a2ea43e61:OoQCy9n5ScWxIROJc7_jmBPbLNQvZl1u  https://sso.example.com
/uas/oauth2/token -d grant_type=password -d "scope=openid 45625ffa-912d-427d-bf27-2f36fe1c4ecf" -d
username=apiuser -d password=apipass
```

The access token is the value of the access_token in the returned json.

The environment variable BEARER can be set to this value and is referred to in future calls as %BEARER%.

```
set
BEARER=eyJjbGllbnRfaWQiOiI5OWI1ZDg1Yi0XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXetc
```

To register an OAuth2 (or OpenID Connect) application using the SSO Management API which has the following settings:

- add new OAuth2 application called `Example5` in the site Helsinki
- add Technical Name (`Example5`)
- add Description
- add UI template name `brandA`
- add Contact e-mail
- add Status (enabled)
- add Force reauthentication (active)
- add Prevent sso after use (active)
- add Application compatibility flags: `MobileConnectLoginHint`

```
curl -v -H "Authorization: Bearer %BEARER%" https://sso.example.com/sso-api/site/Helsinki -d "type=application"
-d "name=Example5" -d "enabled=true" -d "mail=a@example.com" -d "configuration=Compatibility
MobileConnectLoginHint" -d
"configuration=ForceAuthn true" -d "configuration=OneTimeUse true" -d "description=hello world" -d
"template=brandA" -d "mail=a@example.com"
```

Metadata describing the connecting service must be applied to the application. For the following metadata:

```
{"redirect_uris":["http://address/index.html"],"grant_types":["authorization_code"]}
```

the following command would be used. This JSON metadata needs to escaped within the curl command:

```
curl -v -X POST -H "Authorization: Bearer %BEARER%" -H "Content-Type: application/json" https://sso.example.com
/sso-api/application/Helsinki/Example5/$attribute/metadata -d "{\"redirect_uris\":[\"https://address/index.
html\"],\"grant_types\":[\"authorization_code\"]}"
```

A client_id and secret will be returned in the JSON response to this request.

In rare cases, such as GSMA MobileConnect, the client_id and secret is dictated and provided by a third-party. To upload the metadata provided by someone else containing client_id and secret such as

```
{"redirect_uris":["http://address/index.html"],"grant_types":["authorization_code"],"client_id":"
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx","secret":"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"}
```

the following command would be used. Note the `?policy=keep_client_credentials` option, which will ensure client_id and secret is taken from the metadata and not generated. This JSON metadata needs to escaped within the curl command:

```
curl -v -X POST -H "Authorization: Bearer %BEARER%" -H "Content-Type: application/json" https://sso.
example.com/sso-api/application/Helsinki/Example5/$attribute/metadata?policy=keep_client_credentials -d
"{\"redirect_uris\":[\"https://address/index.html\"],\"grant_types\":[\"authorization_code\"],\"
client_id\":\"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\",\"secret\":\"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\"}"
```

The following command will create a PersistentID mapping in the `Helsinki` site called `PersistentIDExample5`. This is known as an outboundMappingPolicy .

```
curl -v -X PUT -H "Authorization: Bearer %BEARER%" https://sso.example.com/sso-api/outboundMappingPolicy
/Helsinki/PersistentIDExample5 -d "nameIDFormat=urn:oasis:names:tc:SAML:2.0:nameid-format:persistent" -d
"nameValue=PersistentIDFormat UUID"
```

The PersistentID outboundMappingPolicy will now be attached to the Example5 application:

```
curl -v -X PUT -H "Authorization: Bearer %BEARER% " https://sso.example.com/sso-api/application/Helsinki
/Example5/$link/outboundMappingPolicy/Helsinki/PersistentIDExample5
```

To set permissions of which groups may use the applications, set the Allowed to group. This command assumes there is a group called "Example5 Users" in the site "Helsinki".

```
curl -v -X PUT -H "Authorization: Bearer %BEARER% " https://sso.example.com/sso-api/application/Helsinki
/Example5/$link/allowedTo/group/Helsinki/Example5%20Users
```

An Authorization Policy defines which user or session attributes are sent to a connected application. In this example, the Authorization Policy called `AuthPolicyX` in the Helsinki site is connected to the Example5 application. Only one Authorization Policy can be connected to an application - setting a new one will remove the old one.

```
curl -v -X PUT -H "Authorization: Bearer %BEARER%" https://sso.example.com/sso-api/application/Helsinki/Example5
/$link/policy/Helsinki/AuthPolicyX
```

The following command sets the Authentication Methods that are allowed to be used for the connected Application. In this example, `password.1` is enabled.

```
curl -v -X PUT -H "Authorization: Bearer %BEARER%" https://sso.example.com/sso-api/application/Helsinki/Example5
/$link/method/password.1 -d "enabled=true"
```

You can enable an existing application using the following command:

```
curl -v -X PUT -H "Authorization: Bearer %BEARER%" http://sso.example.com/sso-api/application/Helsinki/Example5
-d "enabled=true"
```

You can disable an existing application using the following command:

```
curl -v -X PUT -H "Authorization: Bearer %BEARER%" http://sso.example.com/sso-api/application/Helsinki/Example5
-d "enabled=false"
```

When an application is disabled, a user trying to log in will be shown the error message `AGENT_DISABLED` will be shown. This can be customised in `errors.properties`.

All actions done via the SSO Management API are logged in the ubisecure-sso\ubilogin\logs\sso-api.log file. It is rolled over daily.

This page is an illustrative example only - typically these calls would be made by other scripting languages or orchestration techniques.

Related articles

- Use the Ubisecure SSO Management API with curl
- Add relying party using SSO Management API