Client Credentials Flow

Introduction

OA4MP supports the client credentials flow as of version 6.0. This allows a service client to make requests directly to the token endpoint. By ***service client*** we mean a client that is run as a service and has no user. In OA4MP this is a special trust relationship that is granted to a client and attempts to use this by a non-service client will raise an error.

Creating a service client.

There are two ways.

First. Register a client using the register endpoint, email the system administrators that this is to be a service client and state your case. Remember that a service client has considerable discretion in starting flows since effectively the server trusts it. If you are creating or approving the client as an system administrator, set the **`service_client`** property to true using the CLI.

Second. Using the client management endpoint, you may create a regular, confidential client and set the following parameters as needed

Parameter	Default	Description
is_service_client	false	If **true** this is a service client
service_client_users	*	List of names allowed to be set as the subject. Default is any name is allowed.

You do not need to send a subject or have users. However, you may do that and configure your client to return an id token (using the subject you set as its subject) which may, *e.g.*, invoke scripting or a user lookup on the server and return useful information. One example might be to have a set of robots with fixed names and the returned ID token might have information used for initiating a job flow (such as the grant number associated with this) once an access token is obtained.

The Request

The specification really only has the start of the flow and an optional scope. It can only be initiated by a confidential client (so one with an id and secret or one that uses RFC 7523). In additional to that, OA4MP supports the following parameters, *all of which are optional, as per the specification:*

Parameter	Default	Description
at_lifetime	default	The access token lifetime. Assumed to be in seconds, but units are allowed.
audience	--	Either multiple parameters or a space delimited list of human readable logical names of services where the token will be used
exp	--	Expiration timestamp for this request. If this request cannot be

		serviced before the expiration, a error is raised.
id_token_lifetime	default	If there is an id token requested, this is the lifetime. Assumed to be in seconds, but units are allowed.
iss	--	The issuer of this request. If present it must be the client id
jti	--	A unique identifier for this request
nonce	--	A nonce that the client may use to track its requests
resource	--	URLs of services where the issued token may be used. These are eithjer multiple parameters or a space delimited list
state	--	A string that the client sends which will be returned unaltered to help with its state management
rt_lifetime	default	The refresh token lifetime. Assumed to be in seconds, but units are allowed.
scope	--	Scopes for this request. It may be a space delimited string or multiple parameters
sub	--	The subject of the id token. This implies that the client is configured with the openid scope at creation.

Note that the specification is very clear that in normal operation, only an access token is to be returned. OA4MP will allow you to request an ID token if you include the **openid** scope (which must be in the client's configuration too) and a refresh token if you include the **`offline_access`** scope. Again, the client must be configured to allow refresh tokens or this request is ignored.

RFC 7523 support

If you set public keys for your client, you may make requests using RFC 7523 authentication. It is also possible then to replace the client credentials flow *in toto* using RFC 7523 to request tokens directly (one of its purposes was to make a secure replacement for the client credentials flow). It is strongly urged that if possible you at least make your requests using private key signing.