Token-based authentication

On this page:

If you want to authenticate your application or script with Teamwork Cloud, you can use the following procedures.

Pre-configuring Authentication server

Authentication server implements the OpenID Connect standard with several customizations. To access the OpenID Connect configuration, go to *https://<auth_server_host>:<port>/authentication/.well-known/openid-configuration*.

The Authentication server must be configured to accept new client applications by changing these parameters in **authserver.properties** file:

1. Add URL of the client app to the whitelist, separate URLs with comma: **authentication.redirect.uri.whitelist**. This can be either full URL where users should be redirected back from the Authentication server, or just the beginning of it. Authorization endpoint will not accept redirect uri parameters that cannot be found in the whitelist.
2. Add new client IDs, separated with comma: **authentication.client.ids**. You might need to uncomment this line first. Authorization endpoint will not accept **client_id** parameter that cannot be found in this list.

There are a few deviations from standard OpenID Connect specification:

- When invoking token endpoint, HTTP header X-Auth-Secret with secret must be passed with value from **authserver.properties**, parameter **authentication.client.secret**.
- ID tokens have expiration time (configuration property **authentication.token.expirity**), they must be refreshed through the token endpoint by passing refresh tokens.

To call TWC REST API with generated authentication token, the token should be send in the header of the request:

```
Authorization: Token <received_id_token>
```

Authentication with user interaction

The basic Authorization flow should be as follows:

1. Redirect the user to the AuthServer with HTTP GET parameters:

```
scope=openid
redirect_uri=<your_app_url>
client_id=<your_client_id>
response_type=code
```

2. After user signs in, receive HTTP GET request with code parameter.

3. Send HTTP POST request to the token endpoint of the Authentication server with HTTP header X-Auth-Secret and parameters:

```
scope=openid
redirect_uri=<your_app_url>
client_id=<your_client_id>
grant_type=authorization_code
code=<code_received_after_user_signs_in>
```

4. Receive back the JSON response with ID Token that can be used to authorize with TWC and refresh token that later should be used to refresh ID Token.

5. Refresh the ID Token by sending HTTP POST request to the token endpoint of the AuthServer with HTTP header X-Auth-Secret and parameters:

```
scope=openid
redirect_uri=<your_app_url>
client_id=<your_client_id>
grant_type=refresh_token
refresh_token=<refresh_token_value>
```

Authentication without user interaction

To get a token without user interaction, i.e. use some predefined username and password and make only server-server calls, system needs to send HTTP POST request to the token endpoint of the Authentication server with HTTP headers X-Auth-Secret and parameters:

- Headers:

```
X-Auth-Secret: <secret from authentication.client.secret >
```

Authorization: Basic xxxxxxxxx, where xxxxxx is base64 encoded username:password

- Query parameters:

```
grant_type=client_credentials
client_id=<your_client_id>
```

If your client ID is added into the **authentication.client.permanent** list, the returned token will have longer expiration time, configured in the parameter **authentication.permanent.token.expirity**.