Refreshing Tokens

Desktop/Mobile App Users

For desktop/mobile apps, the user must be properly configured on Whitespace. If in doubt, please check the Admin portal/consult one of your organisation's Whitespace Admins.

1. Retrieve the user's registered Whitespace email.
2. Call `GET {ROOT}/auth/apiLogin?email={email}...` where `{ROOT}` is the root URL of the Whitespace environment to provide a token for, i.e. https://sandbox.whitespace.co.uk. This call returns a `redirect` URL.
3. Open a WebView to this URL and wait for it to redirect back to the login screen.
4. Have the user enter their password/complete MFA as required.
5. Once login is completed, the user is redirected to a new URL. Extract the `code` and `state` parameters from this URL
6. `POST` these parameters to `/auth/apiLogin?email={email}` – an example payload is shown directly below. This operation returns a `token` to then connect to the API with. For an example payload, see item (6) in the 'Web Version' checklist. The token lasts for one hour unless refreshed.

   ```
   {"code" = "A.0Tghsfgj….", "state" = "MS"}
   ```

7. To refresh the token, call `/auth/apiRefresh` every 15-45 minutes with the `Authorization: Bearer` header set. This returns a new `token` with another full one-hour duration. This call will return an error if the token has already expired, or is performed when the current token is less than 10 minutes old.

Web Users

For Web site implementations, the user will need to be redirected to the correct URL after authorisation. As a first step before generating tokens, you need to need inform us of this URL and request a `client_id` token. We will configure the token to trigger the correct redirect, and send it to you.

As per Desktop/Mobile Apps, the user must be properly configured on Whitespace. If in doubt, please check the Admin portal/consult one of your organisation's Whitespace Admins.

0. One-time prerequisite: Obtain a `client_id` as described above.
1. Retrieve the user's registered email.
2. Call `GET {ROOT}/auth/apiLogin?email={email}&client_id={client_id}...` where `{ROOT}` is the root URL of the Whitespace environment to provide a token for, i.e. https://sandbox.whitespace.co.uk. This call returns a `redirect` URL.
3. Open this URL and wait for it to redirect back to the login screen.
4. Have the user enter their password/complete MFA as required.
5. Once login is completed, the user is redirected to the URL associated with the `client_id`. Extract the `code` and `state` parameters from this URL

6. `POST` these parameters to `/auth/apiLogin?email={email}` – an example payload is shown directly below. This operation returns a `token` to then connect to the API with. The token lasts for one hour unless refreshed.

```
{"code" = "A.0Tghsfgj….", "state" = "MS"}
```

An example response is:

```
{
    "token": "eyJ0eXAiOiJKV……. ",
    "userIDs": [
        "MU7D7E342A-2649-4DA7-96AC-EBBD6D893422",
        "MU9FCDFG4E-7694-420A-BE61-2ABEDB3E8F52",
        "MU11678FB6-274C-4E58-B7A9-D85A00345623",
    ]
}
```

7. To refresh the token, call `/auth/apiRefresh` every 15-45 minutes with the `Authorization: Bearer` header set. This returns a new `token` with another full one-hour duration. This call will return an error if the token has already expired, or is performed when the current token is less than 10 minutes old.

SUMO Users

When the user has Whitespace logins for in multiple corporate identities, an additional header needs to be passed to identify the unique account being used. This is passed in the header in the form `"UserID: MUxxx"` where MU is the prefix of the unique 36-character identifier string assigned to the user for that corporate identity.

To find out a user's `{uniqueID}` MU string for a specific corporate identity, make a call to `$API/shared/corporate`. The response includes details of every user on the Platform. You can then filter or search this output for the `"members"` arrays that the user's name appears in. The `"members"` array holds the user's `"uniqueID": "MU... "` key-value pair, and is part of a larger array that also includes the `{_id}` key-value pair which clearly identifies the corporate identity associated with that `uniqueID`.

Please note that each given MU value is permanent. If you make a note of it, it can then be used for that user forever.

If the UserID header is not supplied or is incorrect, the call/error response is:

```
curl -H "Authorization: Bearer $TOKEN" $API/risks

{
    "reason": "Invalid WSAUTH: Valid UserID must be supplied",
    "error": true
    }
```

Setting the value and then using it in the UserID heading will work:

```
export UIMU=MU00000000-0000-0000-0000-000000000000
curl -H "Authorization: Bearer $TOKEN" -H "UserID:$UIMU" $API/risks
```

```
[
    {
        "ContractType": "Non-Bulking Line Slip",
        "InsuredName": "AV Facility",
        "IsFacility": true,
        "Status": "Signed",
        "channels": [
            "gailbkr_ALL",
            "mcneillieuwr_ALL"
            ],
        ]
    ]
```

To then authorise the SUMO user for other organisational identities, simply pass a different `UserID`:

```
export UIMU=MU00000000-0000-0000-0000-000000000000
curl -H "Authorization: Bearer $TOKEN" -H "UserID:$UIMU" -s $API/risks | jq '. | length'
# 1528 risks

export UIMU2=MU11111111-1111-1111-1111-111111111111
curl -H "Authorization: Bearer $TOKEN" -H "UserID:$UIMU2" -s $API/risks | jq '. | length'
# 439 risks
```