REST API

- Key Concepts
- Using the API
    - Logging In (Creating a Refresh Token)
    - Access Tokens
    - Logging Out (Deleting a Refresh Token)
    - Requesting Resources
- Web SSO
- Troubleshooting

Did you know?

The full documentation of the current REST services is available in our external developer site.
Click here to access it.

Key Concepts

The REST API is available under the `/api` namespace. For example, `https://yellowfin.myapp.com/api/stories`

The suite includes RPC calls as well, in the `/api/rpc` namespace.

Every API request requires an **Authorization header**. Its format is
`YELLOWFIN ts=1600224140615 nonce=3370ddc4-37d9-41b9-9f24-ada181fdc4bf token=securityToken`

Component	Description
`YELLOWFIN`	Custom authentication scheme
`ts`	The time in milliseconds from the Unix epoch 00:00:00 UTC on 1 January 1970. This is the current time in the program which calls the API. Every programming language has a way to get the current time in this format.
`nonce`	A random UUID generated by the client.
`token`	A security token used for authenticating the user and authorizing access to the resource.

Every API request requires an **Accept header**.

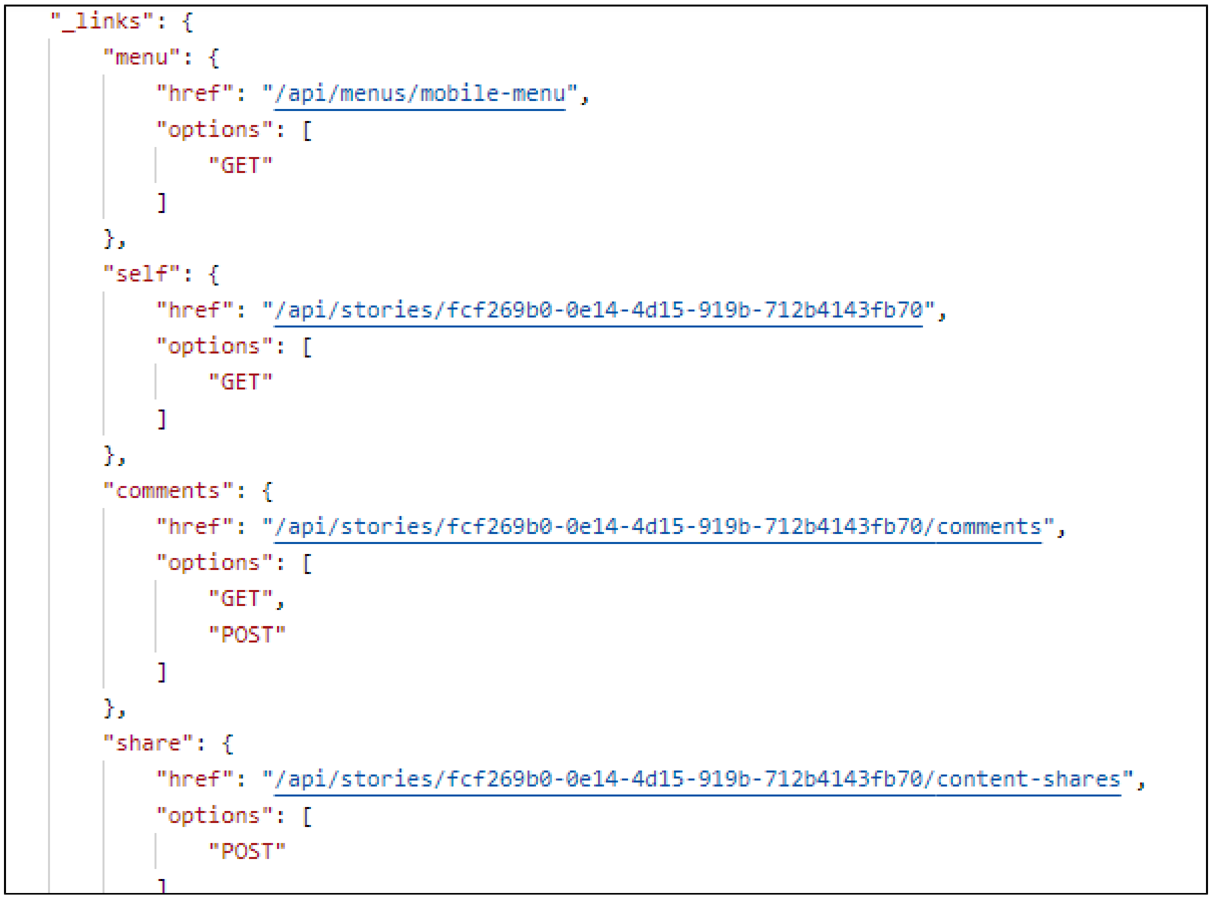
- This header is used to identify the version of the API.
- Its format is specified in the API doc for each endpoint. Usually, it's `application/vnd.yellowfin.api-v`**`1`**`+json`
- The API is backwards compatible. Requests for a v1 resource will work even when the current API version in a Yellowfin instance is v2.

There are two security tokens which are key for consuming the API.

Token	Description
Refresh	This is an opaque security token obtained on login. Refresh Tokens do not expire and may be securely saved in the client application for obtaining access tokens.
Access	This is a JSON Web Token (JWT) which expires after 20 minutes. An access token needs to be sent in the Authorization header of nearly every API request. On expiry, the client application can use the refresh token to get a new access token.

Every API response will have one or more `"_links"` objects.

- Every link represents related resources which the user has access to.
- The client should use the link in the `"href"` attribute to access the resource rather than hard coding it in application code.
- The `"options"` array lists the HTTP methods which the user is authorised to use with the link. For example, the example above tells us that the user can read the comments list (GET) or create a new one (POST). They cannot delete all comments, which is why DELETE is not available in the `"comments"` link.



```
"_links": {
    "menu": {
        "href": "/api/menus/mobile-menu",
        "options": [
            "GET"
        ]
    },
    "self": {
        "href": "/api/stories/fcf269b0-0e14-4d15-919b-712b4143fb70",
        "options": [
            "GET"
        ]
    },
    "comments": {
        "href": "/api/stories/fcf269b0-0e14-4d15-919b-712b4143fb70/comments",
        "options": [
            "GET",
            "POST"
        ]
    },
    "share": {
        "href": "/api/stories/fcf269b0-0e14-4d15-919b-712b4143fb70/content-shares",
        "options": [
            "POST"
```

Using the API

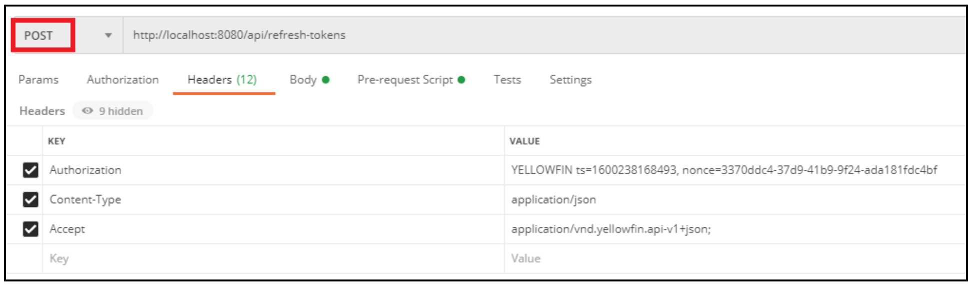
REST API calls may be grouped into the following categories:

1. Logging in — Creating a new refresh token.
2. Access tokens — Used to authorise a user access to REST API resources.
3. Logging out — Deleting a Refresh Token.
4. Requesting Resources — Actually retrieving data using the REST API.

Logging In (Creating a Refresh Token)

Rather than a session, a refresh token is used to identify a user. A consumer must create a refresh token and obtain an access token before they can use other REST endpoints. Creating a refresh token can be thought of as a login process.

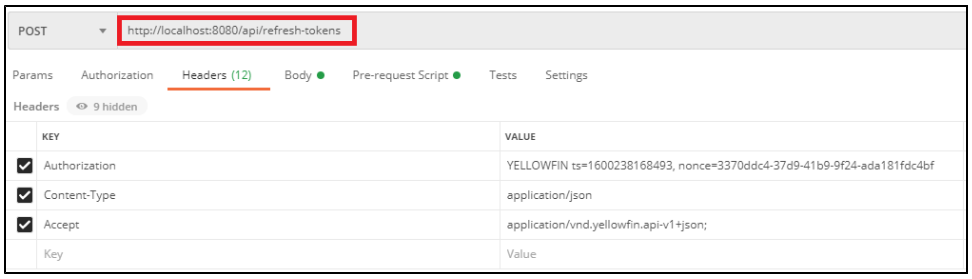
1. Use the HTTP operation POST. Requests that create any kind of resource will always use a POST operation. In this case, a refresh token is being created.



POST http://localhost:8080/api/refresh-tokens

	KEY	VALUE
✓	Authorization	YELLOWFIN ts=1600238168493, nonce=3370ddc4-37d9-41b9-9f24-ada181fdc4bf
✓	Content-Type	application/json
✓	Accept	application/vnd.yellowfin.api-v1+json;
	Key	Value

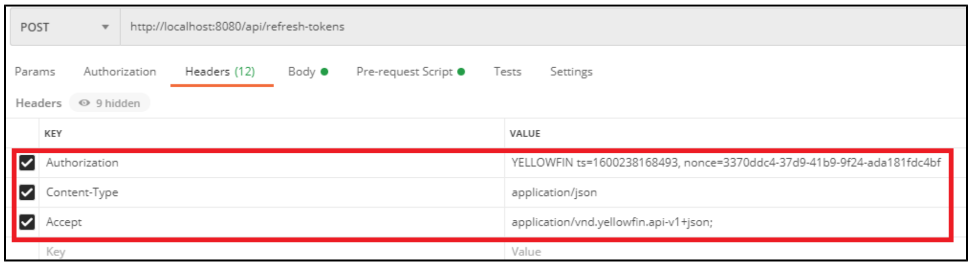
2. Enter the URL of the refresh token endpoint. A valid URL will always have either a name (eg, http://yellowfin.myapp.com/api/...) or an IP address (eg, http://127.0.0.1/api/...). It may have a port specified (eg, http://yellowfin.myapp.com:**8080**/api/...).



http://localhost:8080/api/refresh-tokens

KEY	VALUE
✓ Authorization	YELLOWFIN ts=1600238168493, nonce=3370ddc4-37d9-41b9-9f24-ada181fdc4bf
✓ Content-Type	application/json
✓ Accept	application/vnd.yellowfin.api-v1+json;
Key	Value

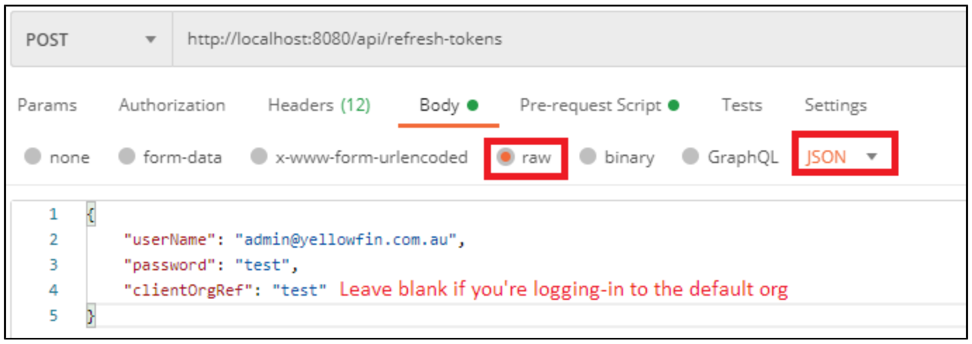
3. Set the mandatory request headers. Refer to the REST API for a full list of headers required to make an API request.



http://localhost:8080/api/refresh-tokens

KEY	VALUE
✓ Authorization	YELLOWFIN ts=1600238168493, nonce=3370ddc4-37d9-41b9-9f24-ada181fdc4bf
✓ Content-Type	application/json
✓ Accept	application/vnd.yellowfin.api-v1+json;
Key	Value

4. The request body contains a JSON representation of a username and password. Make sure that the body is sent as raw JSON.

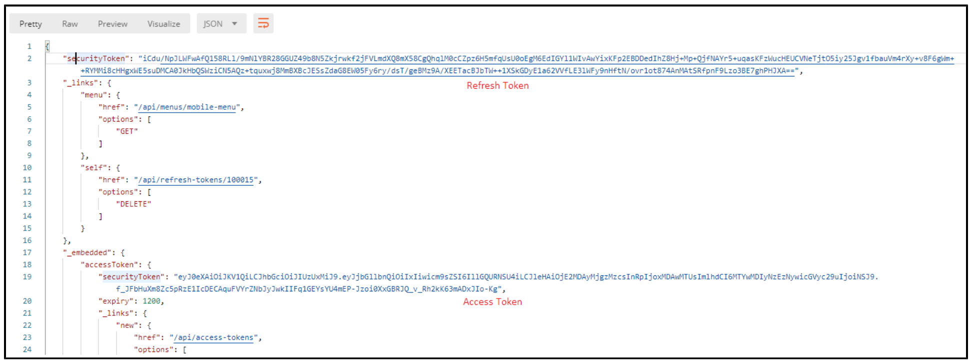


http://localhost:8080/api/refresh-tokens

[✓ raw] [✓ JSON]

```
{
    "userName": "admin@yellowfin.com.au",
    "password": "test",
    "clientOrgRef": "test"  Leave blank if you're logging-in to the default org
}
```

The response of this request will contain the newly-created refresh token, and under the `_embedded property`, an access token.



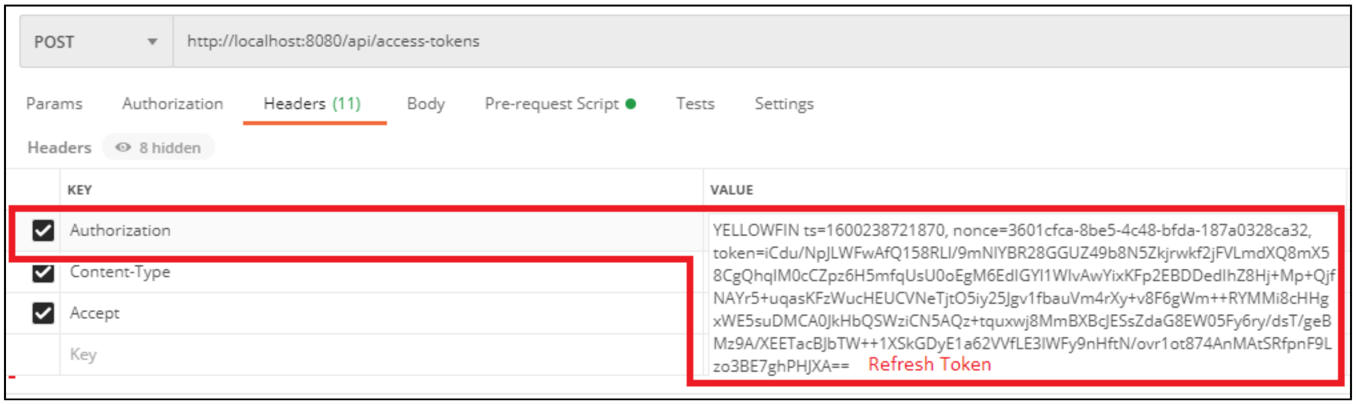
```
{
    "securityToken": "iCdu/NpJLWFwAfQ158RL1/9mN1YBR28GGUZ49b8N5Zkjrwkf2jFVLmdXQ8mX58CgQhqlM0cCZpz6H5mfqUsU0oEgM6EdIGYl1WIvAwYixKFp2EBDDedIhZ8Hj+Mp+QjfNAYr5+uqasKFzWucHEUCVNeTjtO5iy25Jgv1fbauVm4rXy+v8F6gWm+
        +RYMMi8cHHgxWE5suDMCA0JkHbQSWziCN5AQz+tquxwj8MmBXBcJESsZdaG8EW05Fy6ry/dsT/geBMz9A/XEETacBJbTW++1XSkGDyE1a62VVfLE31WFy9nHftN/ovr1ot874AnMAtSRfpnF9Lzo3BE7ghPHJXA==",
    "_links": {
        "menu": {
            "href": "/api/menus/mobile-menu",
            "options": [
                "GET"
            ]
        },
        "self": {
            "href": "/api/refresh-tokens/100015",
            "options": [
                "DELETE"
            ]
        }
    },
    "_embedded": {
        "accessToken": {
            "securityToken": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJjbGllbnQiOiIxIiwicm9sZSI6IllGQURNSU4iLCJleHAiOjE2MDAyMjgzMzcsInRpIjoxMDAwMTUsImlhdCI6MTYwMDIyNzEzNywicGVyc29uIjoiNSJ9.
                f_JFbHuXm8Zc5pRzE1IcDECAquFVYrZNbJyJwkIIFq1GEYsYU4mEP-Jzoi0XxGBRJQ_v_Rh2kK63mADxJIo-Kg",
            "expiry": 1200,
            "_links": {
                "new": {
                    "href": "/api/access-tokens",
                    "options": [
```

⚠ The client application should securely store these tokens. It should also store the `"self"` link as it will be needed for logging out.

Access Tokens

Creating an access token is a very similar process to creating a refresh token. To create one:

- use the POST operation
- use the URL of the access token endpoint
- use the same headers as the refresh token request
    - the Authorization header must specify a refresh token, with a property named `token`



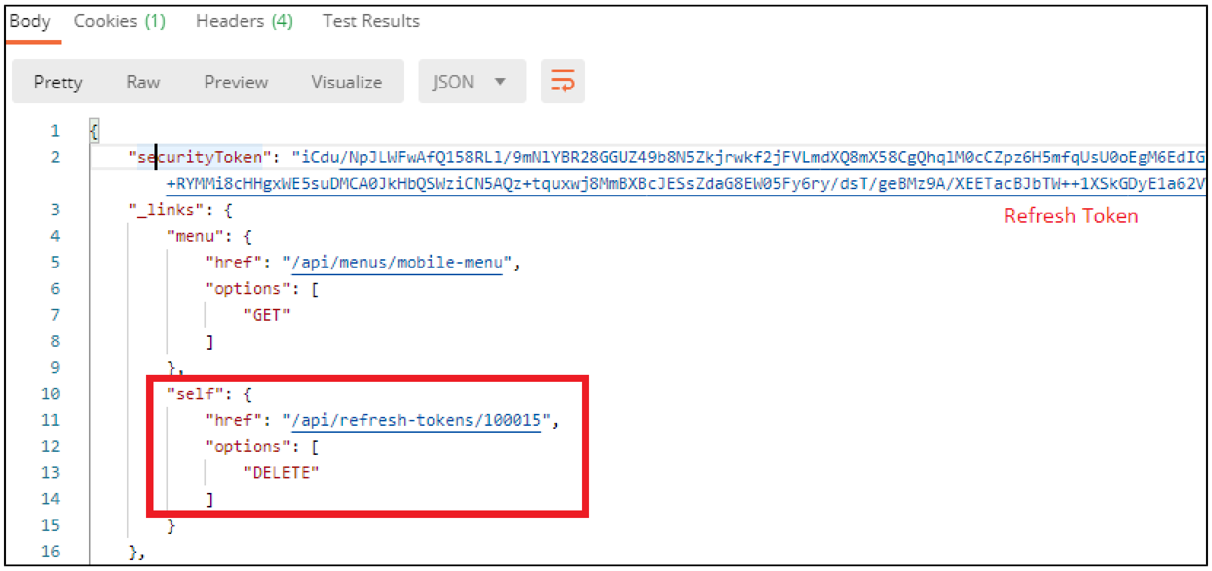
```
POST http://localhost:8080/api/access-tokens
```

KEY	VALUE
Authorization	YELLOWFIN ts=1600238721870, nonce=3601cfca-8be5-4c48-bfda-187a0328ca32, token=iCdu/NpJLWFwAfQ158RLI/9mNIYBR28GGUZ49b8N5Zkjrwkf2jFVLmdXQ8mX58CgQhqIM0cCZpz6H5mfqUsU0oEgM6EdIGYI1WlvAwYixKFp2EBDDedIhZ8Hj+Mp+QjfNAYr5+uqasKFzWucHEUCVNeTjtO5iy25Jgv1fbauVm4rXy+v8F6gWm++RYMMi8cHHgxWE5suDMCA0JkHbQSWziCN5AQz+tquxwj8MmBXBcJESsZdaG8EW05Fy6ry/dsT/geBMz9A/XEETacBJbTW++1XSkGDyE1a62VVfLE3IWFy9nHftN/ovr1ot874AnMAtSRfpnF9Lzo3BE7ghPHJXA==
Content-Type	
Accept	
Key	

The refresh token response provides an access token to make it easier to start consuming the API after login.

Logging Out (Deleting a Refresh Token)

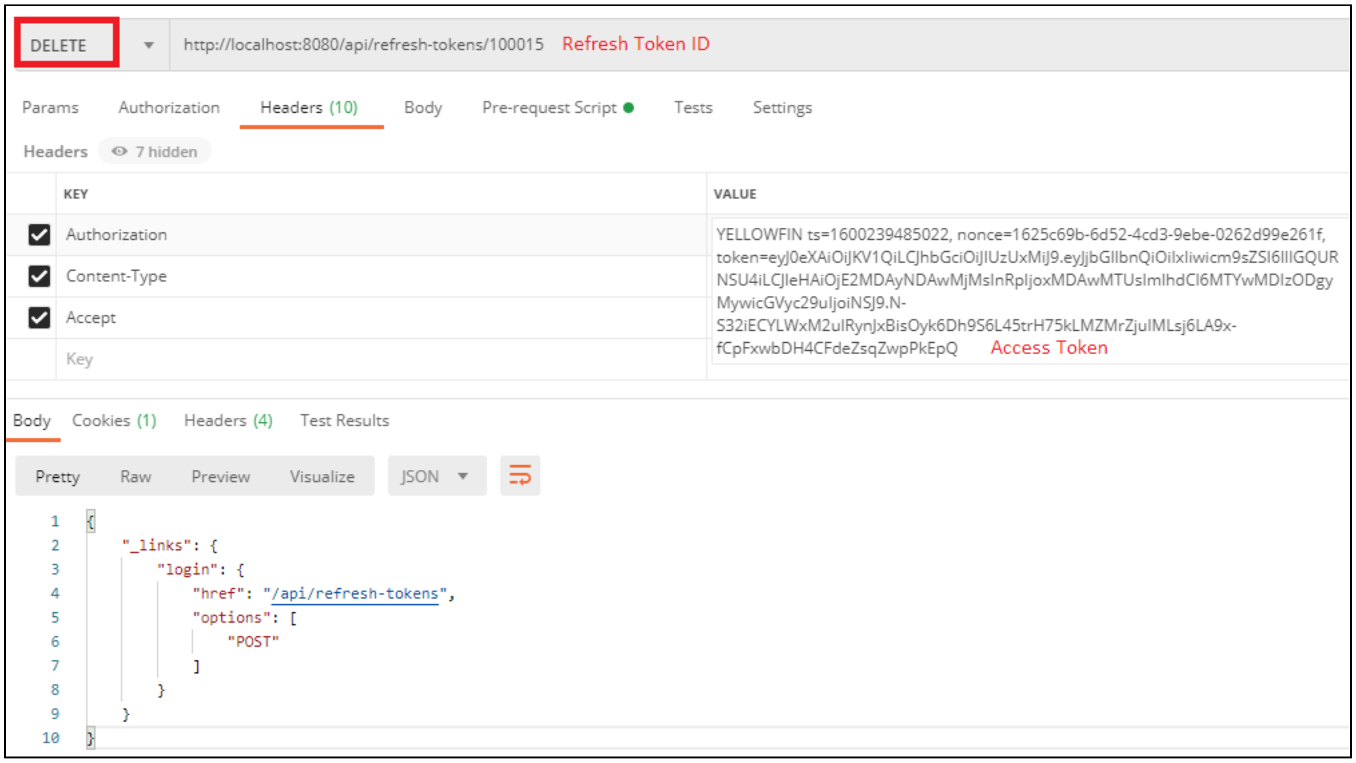
The response of the POST/refresh-tokens request will contain the information required to effectively "log out" of the REST API — a call to delete that refresh token. The response of the POST/refresh-tokens request contains a `_links` property.



```
{
    "securityToken": "iCdu/NpJLWFwAfQ158RL1/9mN1YBR28GGUZ49b8N5Zkjrwkf2jFVLmdXQ8mX58CgQhqlM0cCZpz6H5mfqUsU0oEgM6EdIG
        +RYMMi8cHHgxWE5suDMCA0JkHbQSWziCN5AQz+tquxwj8MmBXBcJESsZdaG8EW05Fy6ry/dsT/geBMz9A/XEETacBJbTW++1XSkGDyE1a62V
    "_links": {
        "menu": {
            "href": "/api/menus/mobile-menu",
            "options": [
                "GET"
            ]
        },
        "self": {
            "href": "/api/refresh-tokens/100015",
            "options": [
                "DELETE"
            ]
        }
    },
```

The options array in the `"self"` link lists which operations can be performed on the new refresh token. There should only be one — `"DELETE"`. Calling DELETE /refresh-tokens will effectively log the user out of the REST API.

Note that a valid access token is required to perform this operation. It must be included in the `token` property of the Authorization header.



http://localhost:8080/api/refresh-tokens/100015

KEY	VALUE
✓ Authorization	YELLOWFIN ts=1600239485022, nonce=1625c69b-6d52-4cd3-9ebe-0262d99e261f, token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJjbGllbnQiOilxliwicm9sZSI6IlIGQURNSU4iLCJleHAiOjE2MDAyNDAwMjMsInRpljoxMDAwMTUslmlhdCI6MTYwMDIzODgyMywicGVyc29uljoiNSJ9.N-S32iECYLWxM2ulRynJxBisOyk6Dh9S6L45trH75kLMZMrZjuIMLsj6LA9x-fCpFxwbDH4CFdeZsqZwpPkEpQ
✓ Content-Type	
✓ Accept	
Key	

```
{
    "_links": {
        "login": {
            "href": "/api/refresh-tokens",
            "options": [
                "POST"
            ]
        }
    }
}
```

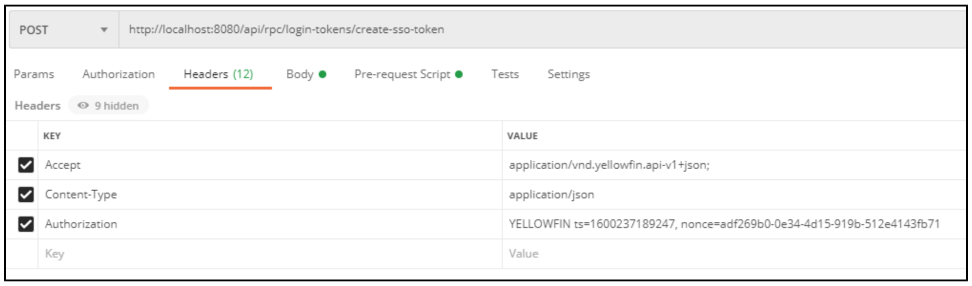
Requesting Resources

To make a resource request, the API client must have a valid access token. **Please consult the API doc for the headers that need to be specified for each endpoint, along with mandatory and optional parameters.**

Web SSO

A popular use-case for the API is Web SSO. A couple of API endpoints are available for generating a login token. The generated token can be used to login to Yellowfin's browser interface. The simplest way to do this is to use the RPC endpoint POST /login-tokens/create-sso-token.

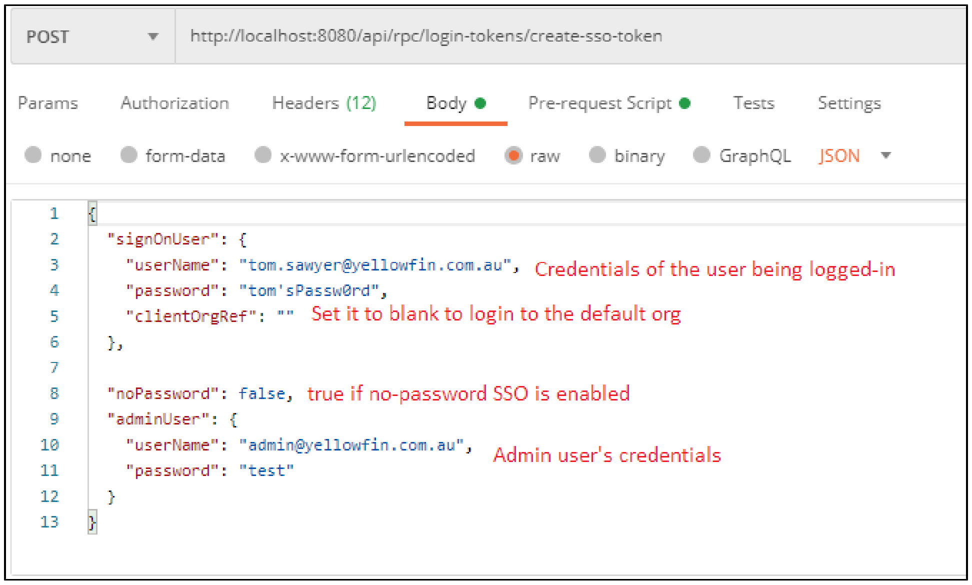
- Use the POST method and set the URL to /login-tokens/create-sso-token
- Requisite headers should be set:



http://localhost:8080/api/rpc/login-tokens/create-sso-token

KEY	VALUE
✓ Accept	application/vnd.yellowfin.api-v1+json;
✓ Content-Type	application/json
✓ Authorization	YELLOWFIN ts=1600237189247, nonce=adf269b0-0e34-4d15-919b-512e4143fb71
Key	Value

- The request body should be set as raw JSON

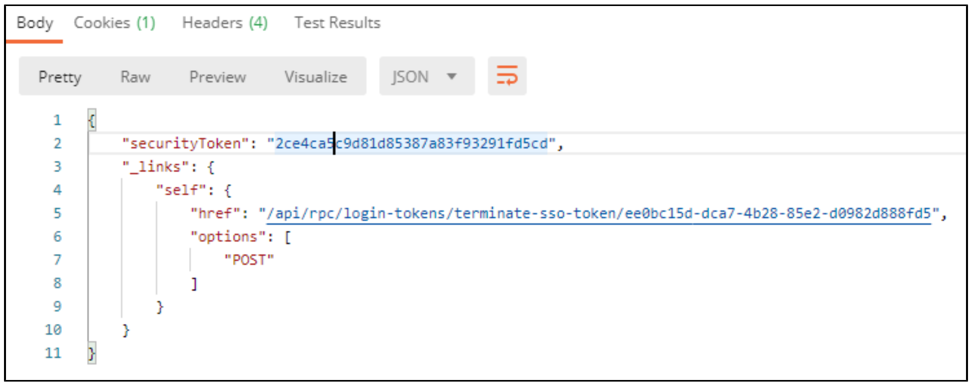


```
http://localhost:8080/api/rpc/login-tokens/create-sso-token
```

```
{
  "signOnUser": {
    "userName": "tom.sawyer@yellowfin.com.au",   Credentials of the user being logged-in
    "password": "tom'sPassw0rd",
    "clientOrgRef": ""   Set it to blank to login to the default org
  },

  "noPassword": false,   true if no-password SSO is enabled
  "adminUser": {
    "userName": "admin@yellowfin.com.au",   Admin user's credentials
    "password": "test"
  }
}
```

- The response contains a login token and the API endpoint for terminating the session.

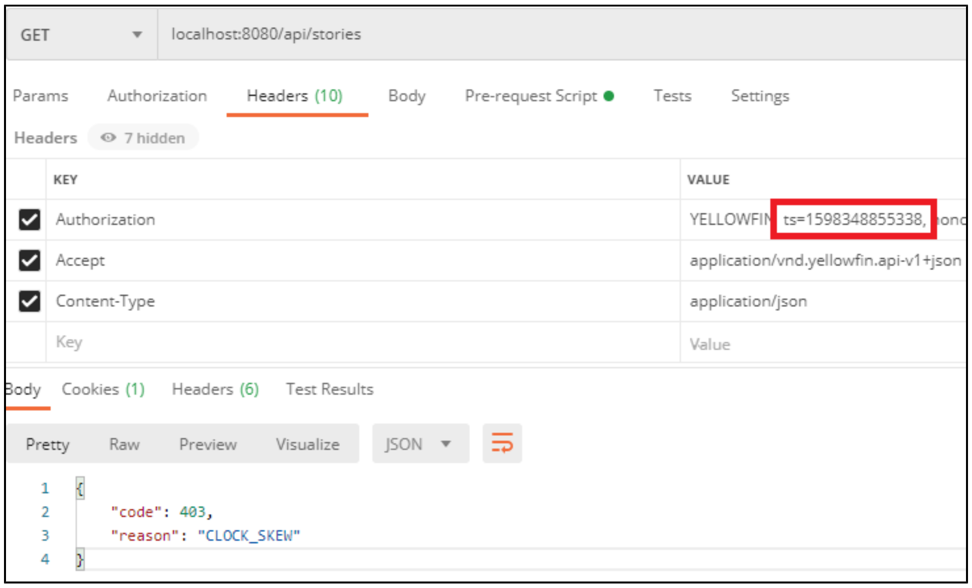


```
{
    "securityToken": "2ce4ca5c9d81d85387a83f93291fd5cd",
    "_links": {
        "self": {
            "href": "/api/rpc/login-tokens/terminate-sso-token/ee0bc15d-dca7-4b28-85e2-d0982d888fd5",
            "options": [
                "POST"
            ]
        }
    }
}
```

- The token may be used for logging into the Yellowfin Web UI or the JavaScript API. See Redirecting to Yellowfin with the Login Token.

Troubleshooting

- Clock Skew — This is one of the most commonly-encountered errors. It is because the timestamp in the Authorization header is not in sync with the server time. There is a +/- 5-minute tolerance but if it falls outside that window, the API will respond with an error.

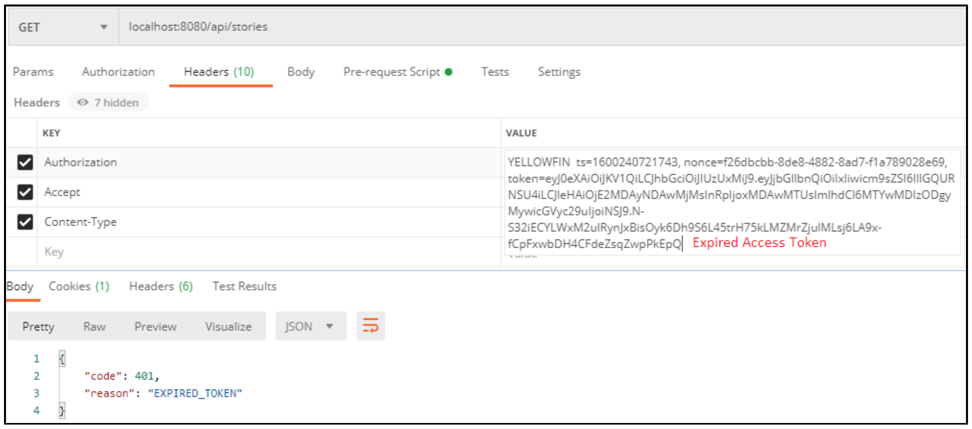


localhost:8080/api/stories

KEY	VALUE
✓ Authorization	YELLOWFIN ts=1598348855338, nonc
✓ Accept	application/vnd.yellowfin.api-v1+json
✓ Content-Type	application/json
Key	Value

```
{
    "code": 403,
    "reason": "CLOCK_SKEW"
}
```

- Token expiry — The API responds with an error when an expired access token is used.

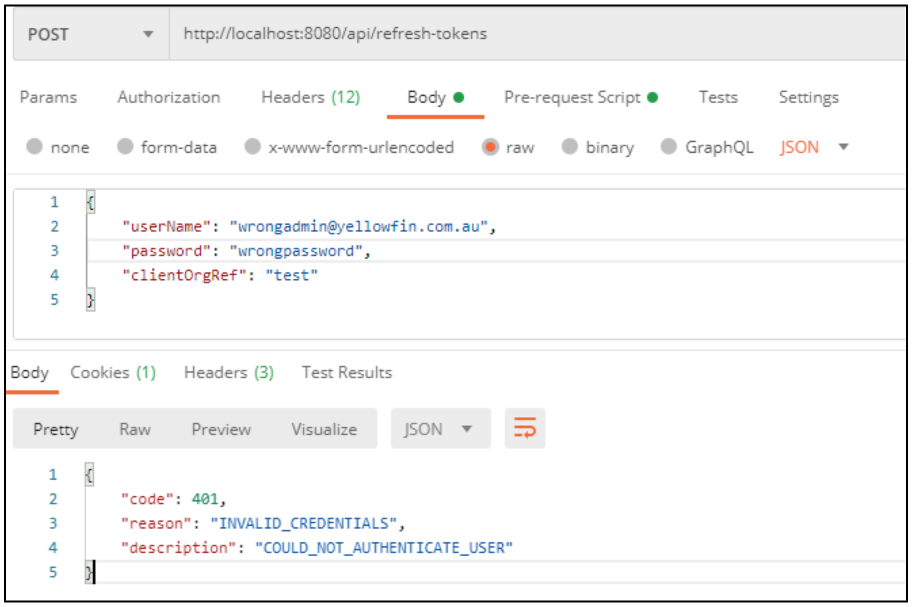


localhost:8080/api/stories

KEY	VALUE
✓ Authorization	YELLOWFIN ts=1600240721743, nonce=f26dbcbb-8de8-4882-8ad7-f1a789028e69, token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJjbGllbnQiOiIxIiwicm9sZSI6IlIGQURNSU4iLCJleHAiOjE2MDAyNDAwMjMsInRpIjoxMDAwMTUsImlhdCI6MTYwMDIzODgyMywicGVyc29uIjoiNSJ9.N-S32iECYLWxM2uIRynJxBisOyk6Dh9S6L45trH75kLMZMrZjuIMLsj6LA9x-fCpFxwbDH4CFdeZsqZwpPkEpQ
✓ Accept	
✓ Content-Type	
Key	

```
{
    "code": 401,
    "reason": "EXPIRED_TOKEN"
}
```

- Authentication failure — This could occur because of an invalid username or password.



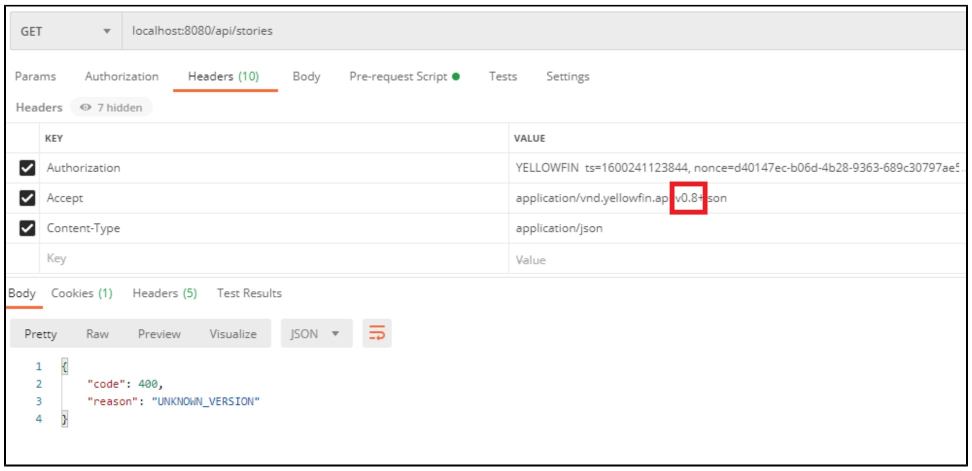
POST http://localhost:8080/api/refresh-tokens

```
{
    "userName": "wrongadmin@yellowfin.com.au",
    "password": "wrongpassword",
    "clientOrgRef": "test"
}
```

Response Body:

```
{
    "code": 401,
    "reason": "INVALID_CREDENTIALS",
    "description": "COULD_NOT_AUTHENTICATE_USER"
}
```

- Unknown version — If an incorrect version of the API is specified in the `Accept` header.



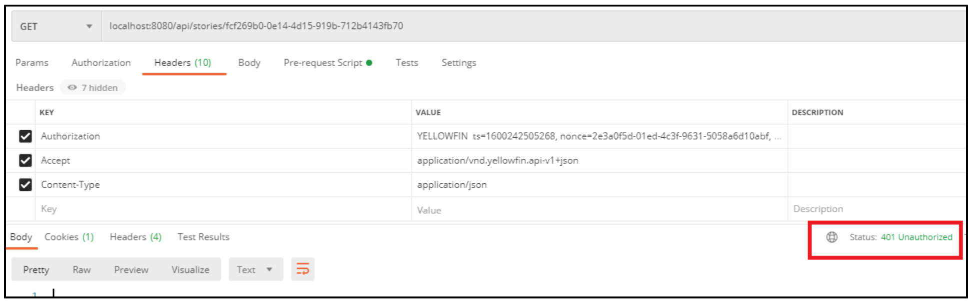
GET localhost:8080/api/stories

	KEY	VALUE
✓	Authorization	YELLOWFIN ts=1600241123844, nonce=d40147ec-b06d-4b28-9363-689c30797ae5
✓	Accept	application/vnd.yellowfin.ap v0.8+json
✓	Content-Type	application/json

Response Body:

```
{
    "code": 400,
    "reason": "UNKNOWN_VERSION"
}
```

- Licencing error — Content services such as GET/stories/uuid, are only available when a server licence is present. If not, the API will return a 401 Unauthorized error.



GET localhost:8080/api/stories/fcf269b0-0e14-4d15-919b-712b4143fb70

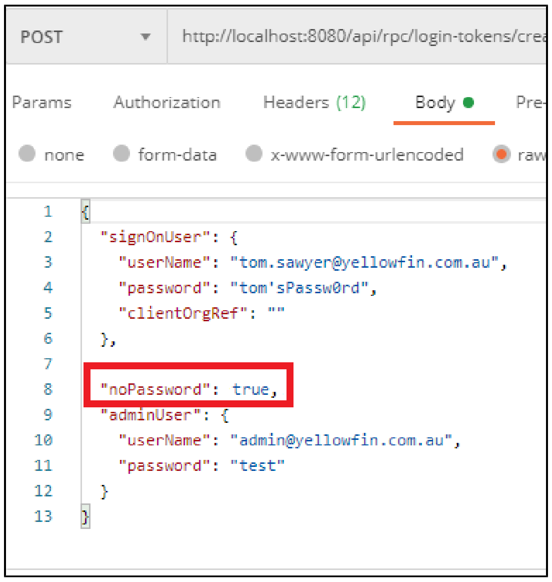
	KEY	VALUE	DESCRIPTION
✓	Authorization	YELLOWFIN ts=1600242505268, nonce=2e3a0f5d-01ed-4c3f-9631-5058a6d10abf, ...	
✓	Accept	application/vnd.yellowfin.api-v1+json	
✓	Content-Type	application/json	

Status: 401 Unauthorized

- CORS — This is generally not a problem for the REST API because CORS applies only to browsers. A web browser is not a recommended REST client as it is not easy to securely store tokens.

- SSO Errors — Ensure that credentials and org reference are correct. If `noPassword` authentication is being used, ensure that it has been enabled on the server. This is done by inserting a record into the Configuration table **and restarting Yellowfin**.

  INSERT INTO Configuration values (1, 'SYSTEM', 'SIMPLE_AUTHENTICATION', 'TRUE');


POST http://localhost:8080/api/rpc/login-tokens/crea

```
{
  "signOnUser": {
    "userName": "tom.sawyer@yellowfin.com.au",
    "password": "tom'sPassw0rd",
    "clientOrgRef": ""
  },

  "noPassword": true,
  "adminUser": {
    "userName": "admin@yellowfin.com.au",
    "password": "test"
  }
}
```

- Error 500 Internal Server Error — This is a generic error message which indicates that something went wrong on the server. Contact support with the error trace in the server logs for more information.

The full documentation of the current REST services is available in our external developer site. Click here to access it.