OSCAR: OAuth Notes

Document History

2016-03-09: Created - Simon Diemert

DISCLAIMER

This is a working document that represents the author's efforts to understand the use of OAuth in OSCAR EMR. It is by no means a complete description of OAuth in OSCAR, and **should not** be used as an authoritative source of information until the details are confirmed.

Purpose

The purpose of this document is to describe the use of OAuth in the OSCAR EMR system. The intended audience are developers of OSCAR or OSCAR Apps who wish to access the RESTful interface problematically.

Background

This section provides background information to orient the reader.

OSCAR RESTful API

OSCAR (14) 15 introduced a RESTful Application Programming Interface (API) to facilitate changes to the user interface and to support access to other applications. The current configuration of OSCAR (as of 2016-03-09) exposes two different RESTful APIs; these use the same underlying Java source code, but use different authentication mechanisms:

- `/ws/services/<path>?<query` uses OAuth as an authentication mechanism. Only those with a valid access token will have requests fulfilled by this service.
- `/ws/rs/<path>?<query>` uses a session based authentication mechanism, this service is used primarily by the "new" OSCAR (14) 15 user interface.

The configuration for both of these services can be found in either:

- `oscar/src/main/resources/spring_ws.xml` (`/ws/rs`)
- `oscar/src/main/resources/ApplicationContextREST.xml` (`/ws/services/`)

The remainder of this document will focus on using OAuth of access OSCAR data via `/ws/services` .

OAuth

Open standard for AUTHentication (OAuth) is a authentication mechanism that allows a third party of access data without exposing the details of their authentication. OAuth is used extensively by popular Internet services like Google, Facebook, and Twitter to allow third party applications to access user data.

OAuth has a number of versions and various sub-protocols, OSCAR uses OAuth 1.0a.

A number of great descriptions of various OAuth protocols can be found online:

- **Basic Introduction:** http://hueniverse.com/oauth/
- **OAuth Bible:** http://oauthbible.com/

Other useful tools:

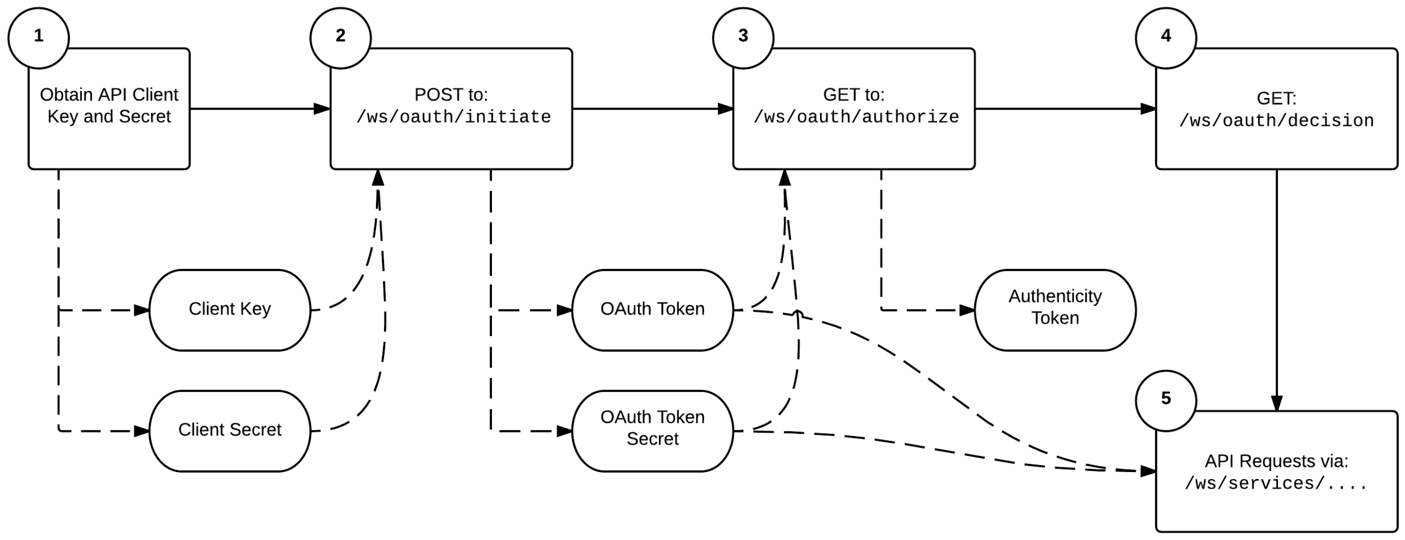
- **Postman:** A Google Chrome application for making requests to REST APIs. Great for exploring the API.
  - https://www.getpostman.com/
  - https://www.getpostman.com/docs/helpers
- **RAML:** A tool for designing and documenting REST APIs
  - http://raml.org/
  - https://github.com/mulesoft/raml-for-jax-rs

Accessing OSCAR via OAuth

This section walks through the steps required to authenticate an application via OAuth and use OSCAR's RESTful web service. The Postman is used to demonstrate each step.

Overview

The following diagram describes the general steps for authenticating against OSCAR's OAuth system:


1
Obtain API Client Key and Secret
2
POST to: /ws/oauth/initiate
3
GET to: /ws/oauth/authorize
4
GET: /ws/oauth/decision
Client Key
Client Secret
OAuth Token
OAuth Token Secret
Authenticity Token
5
API Requests via: /ws/services/....

The remained of this section discusses each of these steps in detail.

1) Obtain Client Key and Secret

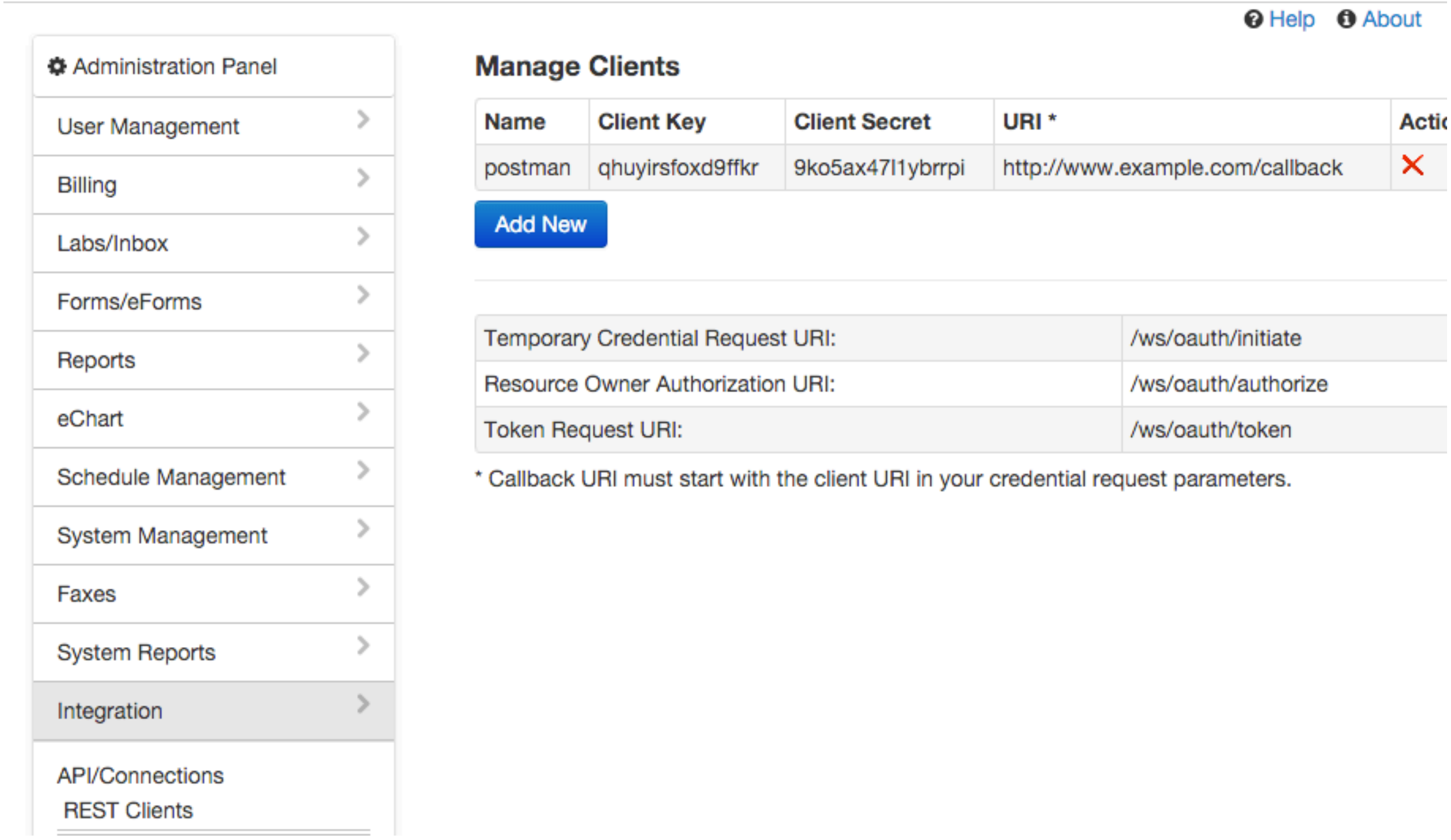
In order to make a request you (or your users) must register their application with OSCAR. This is done by creating a **client key** and **client secret** pair. Your application will uses these to initiate authentication with OSCAR.

Note: Standard OAuth terminology uses *consumer* to refer to the application that is making requests to OSCAR. In OSCAR this has been called *client*.

To create the client key/secret pair open the Administration Panel of OSCAR and navigate to the "REST Clients" panel:

- Administration -> Integration -> API/Connections -> REST Clients

Select "Add New" and enter the name of your application and a callback URI. Save the resulting client key/secret pair somewhere, or keep this window open. In the end it should look something like:



Manage Clients

Name	Client Key	Client Secret	URI *	Acti
postman	qhuyirsfoxd9ffkr	9ko5ax47l1ybrrpi	http://www.example.com/callback	✕

Temporary Credential Request URI:	/ws/oauth/initiate
Resource Owner Authorization URI:	/ws/oauth/authorize
Token Request URI:	/ws/oauth/token

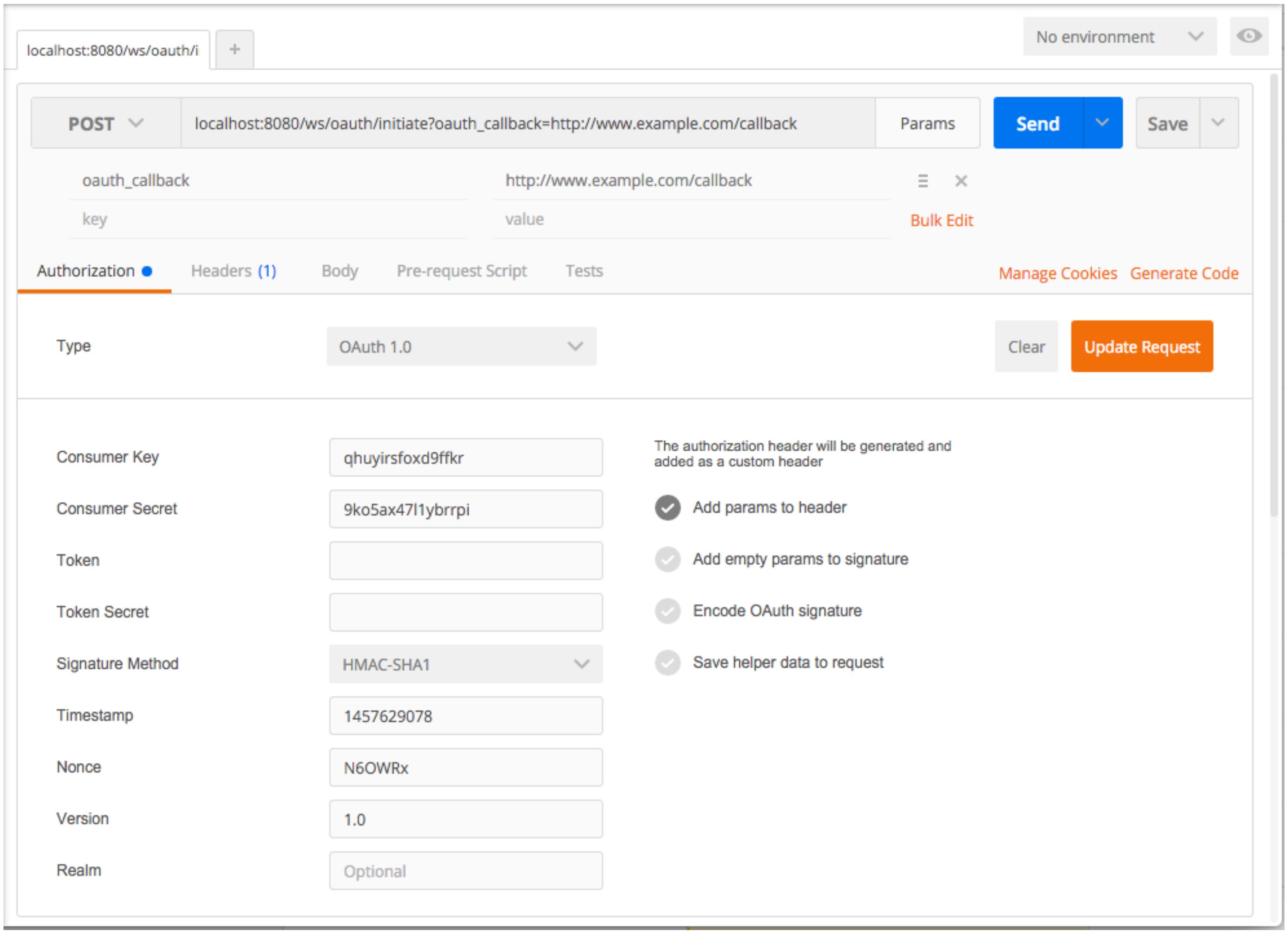
* Callback URI must start with the client URI in your credential request parameters.

2) POST: /ws/oauth/initiate

You must initiate an OAuth connection with OSCAR. Have your client application make a HTTP POST request to: `/ws/oauth/initiate` , you must include:

- **Callback URL:** a callback URL as a HTTP query string parameter. This URL must match the one you provided to OSCAR in Step 1.
    - `/ws/oauth/initiate?oauth_callback=http://example.com/callback`
- **Consumer Key:** the client key from OSCAR in the OAuth 1.0 header.
- **Consumer Secret:**the client secret from OSCAR in the OAuth 1.0 header.

For example:



oauth_callback	http://www.example.com/callback
key	value

Type: OAuth 1.0

Consumer Key	qhuyirsfoxd9ffkr
Consumer Secret	9ko5ax47l1ybrrpi
Token	
Token Secret	
Signature Method	HMAC-SHA1
Timestamp	1457629078
Nonce	N6OWRx
Version	1.0
Realm	Optional

The authorization header will be generated and added as a custom header

- [✓] Add params to header
- Add empty params to signature
- Encode OAuth signature
- Save helper data to request

A comparable unix curl request might be:

```
curl --request POST \
  --url 'http://localhost:8080/ws/oauth/initiate?oauth_callback=http%3A%2F%2Fwww.example.com%2I
  --header 'authorization: OAuth oauth_consumer_key="qhuyirsfoxd9ffkr",
        oauth_signature_method="HMAC-SHA1",
        oauth_timestamp="1457638451",
        oauth_nonce="cb3gUv",
        oauth_version="1.0",
        oauth_signature="CFGV3sBZkauoyjti9TxNCbNlSl4%3D"' \
  --header 'cache-control: no-cache' \
  --header 'postman-token: 8cd4b0b5-9284-3d55-c0d5-2a650a73fb14'
```

This request should return two keys:

- **oauth_token:** An OAuth Token, for example: `d432f767-a1fe-43b6-b2bf-73923fd7ef92`.

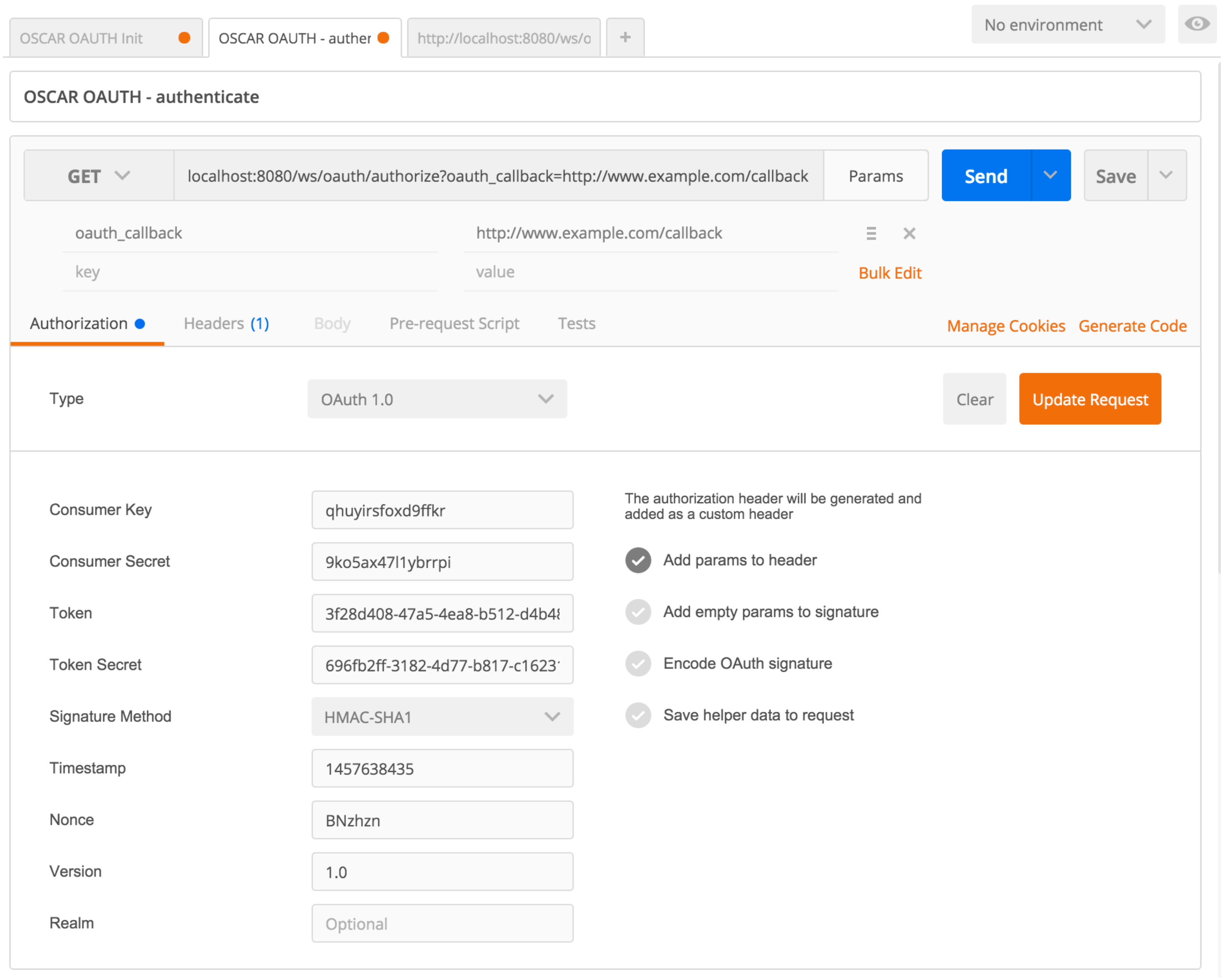
- **oauth_token_secret:** The secret associated with the oauth_token, for example: `b1886b96-287c-49d8-aa9f-92f0353e3230`.

3) GET: /ws/oauth/authorize

You must obtain an **access token** by making a request to `/ws/oauth/authorize`. In addition to the parameters from Step 2 you must include:

- **OAuth Token:** The token returned by OSCAR in step 2.
- **OAuth Token Secret:** The secret returned by OSCAR in step 2.

For example:



OSCAR OAUTH - authenticate

GET localhost:8080/ws/oauth/authorize?oauth_callback=http://www.example.com/callback

oauth_callback	http://www.example.com/callback
key	value

Type: OAuth 1.0

Consumer Key	qhuyirsfoxd9ffkr
Consumer Secret	9ko5ax47l1ybrrpi
Token	3f28d408-47a5-4ea8-b512-d4b4ξ
Token Secret	696fb2ff-3182-4d77-b817-c1623
Signature Method	HMAC-SHA1
Timestamp	1457638435
Nonce	BNzhzn
Version	1.0
Realm	Optional

The authorization header will be generated and added as a custom header

- ✓ Add params to header
- Add empty params to signature
- Encode OAuth signature
- Save helper data to request

A comparable unix curl request might be:

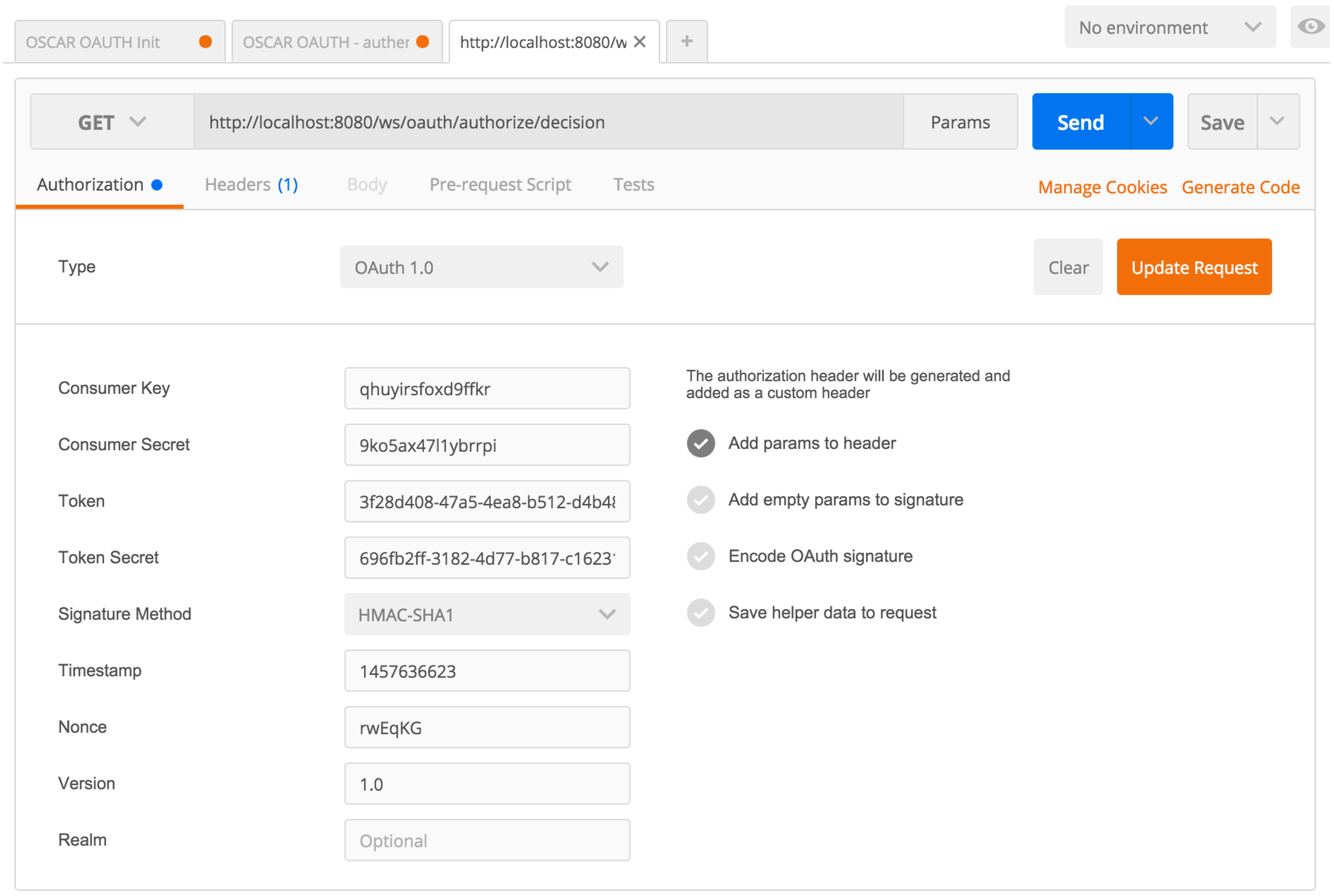
```
curl --request GET \
  --url 'http://localhost:8080/ws/oauth/authorize?oauth_callback=http%3A%2F%2Fwww.example.com%
  --header 'authorization: OAuth oauth_consumer_key="qhuyirsfoxd9ffkr",
        oauth_token="3f28d408-47a5-4ea8-b512-d4b487b3076f",
        oauth_signature_method="HMAC-SHA1",
        oauth_timestamp="1457638435",
        oauth_nonce="BNzhzn",oauth_version="1.0",
        oauth_signature="pqCFnXwBLv38yKS9MGtWtMu7yxU%3D"' \
  --header 'cache-control: no-cache' \
  --header 'postman-token: 1b1424a0-8433-c9fd-1d14-2e2a5a0c05db'
```

This request should return an *access token* (*authenticity token*), for example: `1a3c13d6-acfb-4cd4-96ac-06e4f30c18a0`

4) GET: /ws/oauth/authorize/decision

The authorize response requests that you reply to: `/ws/oauth/authorize/decision`.

For example:



http://localhost:8080/ws/oauth/authorize/decision

Type: OAuth 1.0

Consumer Key	qhuyirsfoxd9ffkr
Consumer Secret	9ko5ax47l1ybrrpi
Token	3f28d408-47a5-4ea8-b512-d4b4
Token Secret	696fb2ff-3182-4d77-b817-c1623
Signature Method	HMAC-SHA1
Timestamp	1457636623
Nonce	rwEqKG
Version	1.0
Realm	Optional

The authorization header will be generated and added as a custom header

✓ Add params to header
Add empty params to signature
Encode OAuth signature
Save helper data to request

A comparable unix curl request might be:

```
curl --request GET \
  --url http://localhost:8080/ws/oauth/authorize/decision \
  --header 'authorization: OAuth oauth_consumer_key="qhuyirsfoxd9ffkr",
        oauth_token="3f28d408-47a5-4ea8-b512-d4b487b3076f",
        oauth_signature_method="HMAC-SHA1",
        oauth_timestamp="1457636623",
        oauth_nonce="rwEqKG",oauth_version="1.0",
        oauth_signature="fggtN1TnJn%2B24puuND0JbkpR9TA%3D"' \
  --header 'cache-control: no-cache' \
  --header 'postman-token: 19e61e2e-92e4-7e45-480f-332fec05d63a'
```

5) Make Requests

Unclear how to do this. Once you have an access token, how do make a request to `/ws/services/xxx` ?

Problems encountered so far:

- How do you associate a providerNo with a request?
    - A lack of providerNo seems to be causing some issues when making the request.