OAuth Authentication

In this section:

Introduction

The OAuth (Open Authorization) authentication protocol enables users to grant third-party applications access to their private resources without revealing login credentials. It also provides a way to restrict the amount of information that can be accessed.

OAuth introduces the *user* role (also known as the *resource owner*) into the traditional client-server authentication model. In the traditional client-server authentication model, the client directly accesses resources hosted on the server. In the OAuth model, the client must first obtain permission from the resource owner before accessing resources from the server. This permission is expressed in the form of a token and matching shared-secret key.

OAuth 2.0 and 1.0a are different implementations and are incompatible. Refer to the OAuth website to learn more: https://oauth.net/2/.

Example Scenario

Assume that a user (resource owner) wants to grant a printing service (client) access to his private photos stored at a photo sharing service (server). Instead of revealing the user's login credentials to the printing service, the user can perform OAuth authentication to grant the printing service permission to access his or her private photos. This would happen in three stages:

1. The printing service requests temporary credentials from the photo sharing service.
2. Once the printing service receives the credentials, it redirects the user to the photo sharing service's OAuth Authorization URL, then the user provides login credentials. Note that the printing service has no visibility into the user's login credentials at this step. Once the user decides to give the printing service access to the private photos, a confirmation verification code is generated.
3. The printing service then exchanges the temporary credentials plus the verification code for an access token. Once the printing service has the access token, they can then obtain and print the user's private pictures from the photo sharing service.

Parasoft Support for OAuth

Parasoft supports OAuth 1.0a and 2.0 security protocols for web server flow and client credentials flow (two-legged scenario). You can configure OAuth authentication settings for OAuth 1.0a in the OAuth Authentication section of the HTTP transport. For OAuth 2.0, the authentication is configured in the REST Client's Resource and Payload tabs.

OAuth 2.0

The OAuth 2.0 RFC specifies different "flows" or "grant types" for authentication:

- Authorization Code (Web Server) Flow
- Client Credentials Flow
- Device Code
- Refresh Token

This documentation describes the two most common flows: the web server (authorization code) and client credentials flows.

See https://oauth.net/2/grant-types/ for additional information about OAuth 2.0 flows.

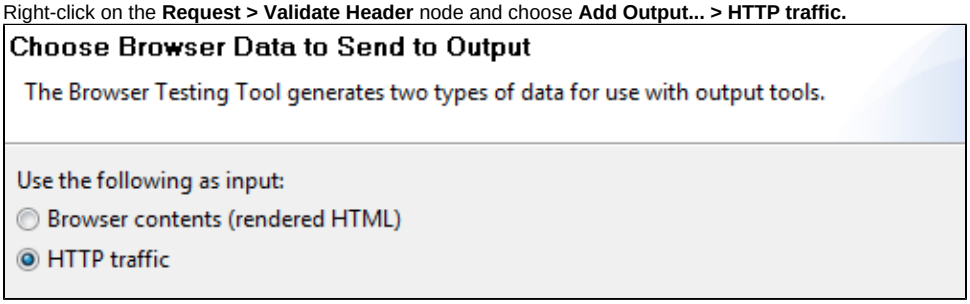
Web Server (Authorization Code) Flow

This grant type is used by confidential and public clients to exchange an authorization code for an access token. After the user returns to the client via the redirect URL, the application will get the authorization code from the URL and use it to request an access token. Refer to the OAuth 2.0 Framework documentation for details about this flow: https://tools.ietf.org/html/rfc6749#section-4.1

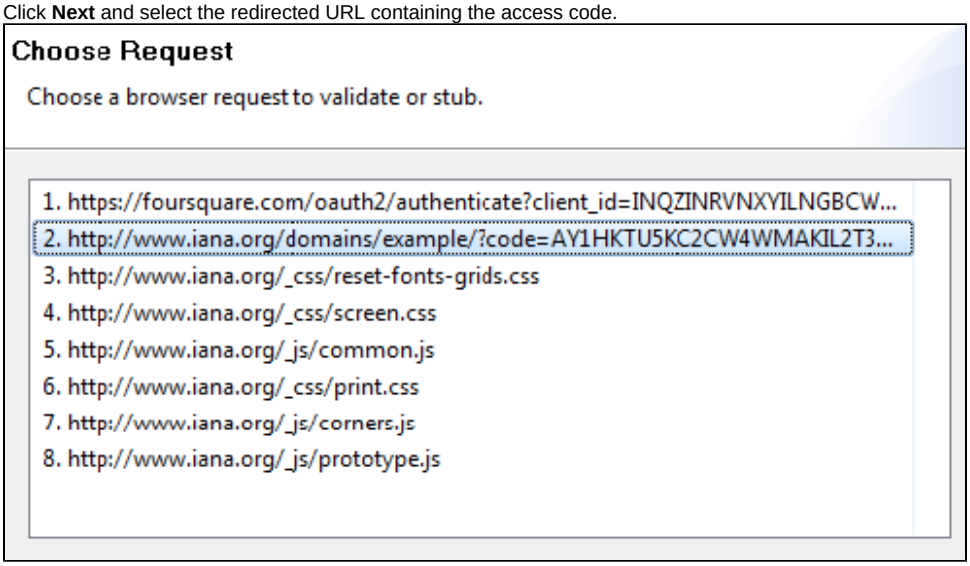
Request Authorization

1. Right-click on your project folder and choose **Add New > Test (.tst) File**.
2. Specify a name for the test and choose **Web > Record web scenario**.
3. Click **Next** and choose **Record new web scenario**.
4. Specify the OAuth URL parameters in the Start Recording From field.
5. Click **Finish** and log into your application. Once authorized, the Service Provider will redirect you to the callback URL with a code as part of a URL parameter.
6. Close the browser to complete the recording.

7. Right-click on the **Request > Validate Header** node and choose **Add Output... > HTTP traffic.**



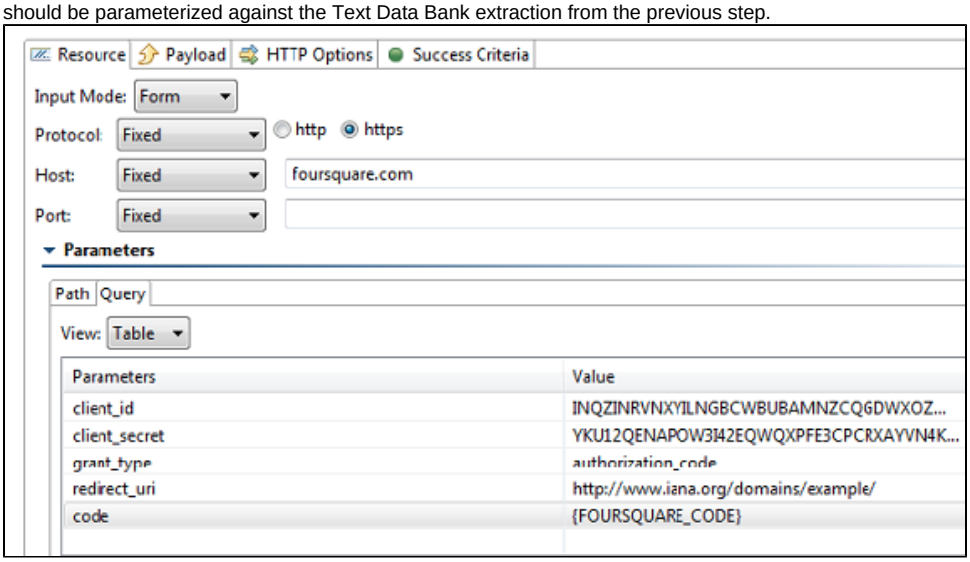
8. Click **Next** and select the redirected URL containing the access code.



9. Click **Next** and choose **Text Data Bank** to extract the code.

Obtain Access Token

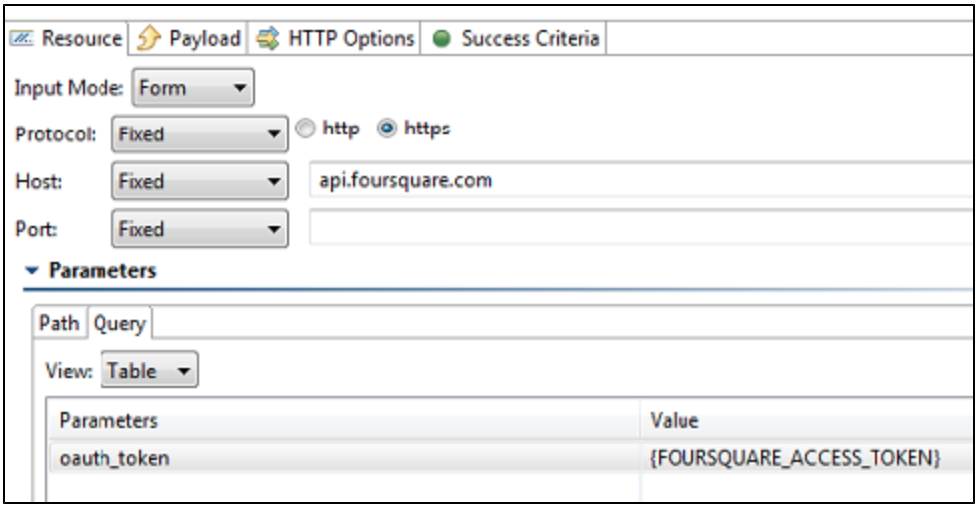
1. Create a new REST Client and provide the URL with the necessary parameters. One of the URL parameters should be called `code`. This value should be parameterized against the Text Data Bank extraction from the previous step.



2. Depending on the Response format, attach the appropriate Data Bank (i.e., Text, JSON) to the REST client and extract the Access Token.

Access Protected Resources

For every REST API call, create a new REST Client and provide it the desired URL with the necessary URL parameters. One of the URL parameter should be called `oauth_token` and will have the value of the access token extracted in the previous test step.



Client Credentials Flow

This grant type is used by clients to obtain an access token outside of the context of a user. This is typically used by clients to access resources about themselves rather than to access a user's resources. Refer to the OAuth 2.0 Framework documentation for details about this flow: https://tools.ietf.org/html/rfc6749#section-4.4

Obtain Access Token

1. Add a REST client to your test suite. For clarity, you can rename the client, e.g., *Obtain Access Token*.
2. Click the **Resource** tab and choose **POST** from the Method drop-down menu.
3. Specify the endpoint of your authorization server in the URL field.
4. Click the **Payload** tab and choose **URL Encoded** from the Payload Format menu.
5. Choose **Table** from the Input mode menu and configure the following parameters:

Parameter	Description	Required

`client_id`
 	
`client_secret`
	These parameters may be used to provide credentials for authenticating with the authorization server. HTTP authentication may be used to authenticate, however, instead of sending these parameters.
	
	If using HTTP authentication, provide credentials under HTTP Options > Security > Http Authentication instead.
		Sometimes

`grant_type`	Value must be `client_credentials`	Yes
`scope`	This parameter is used by the client to request limited access to the application. Specify a comma separated list of values specific to the authorization server. If absent, the authorization server will reject the request or return an access token with a default scope. The authorization server may also return an access token with a different scope than what was initially requested here by the client. See https://oauth.net/2/scope/ for additional information.	Sometimes
`audience`	Specifies the URI of the resource server. This parameter may be required in order to indicate what server the access token will be restricted to access.	Sometimes

6. Run the REST Client and verify the response message. Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

{
  "access_token":"2YotnFZFEjr1zCsicMWpAA",
  "token_type":"Bearer",
  "expires_in":3600,
  "scope":"read_something"
}
```

7. Right-click the Rest Client and choose **Add Output**.
8. Choose **Response > Traffic > JSON Data Bank** and click **Finish**.
9. Create extractions for the `access_token` and `token_type` values. The following section assumes that you have named the data bank columns "access_token" and token_type" to correspond to the extractions.

Access Protected Resource

Your client can now make calls to all protected resources by including the access token extracted from the JSON Data Bank.

1. Add a REST client to your test suite.
2. Click the **Resource** tab and specify your REST call method and endpoint, including any expected parameters.
3. Click the **HTTP Options** tab and choose the **HTTP Headers** settings.
4. Add an HTTP header with name `Authorization` and value of `${token_type} ${access_token}`. The `${token_type}` and `${access_token}` values should match the names of the JSON Data Bank columns created in the previous test. *A single space must separate values*.
5. Run the scenario and verify the expected HTTP request header that was sent. Example Authorization header:
   `Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA`

OAuth 1.0a

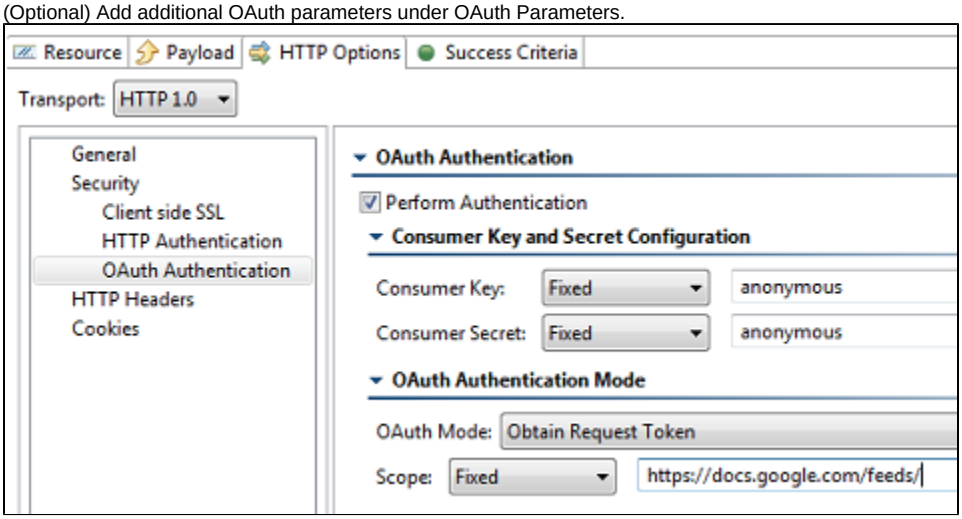
Authenticating against OAuth 1.0a includes the following general steps:

1. Obtain and authorize a request token from the service provider
2. Exchange the request token for an access token
3. Sign OAuth requests to access protected resources

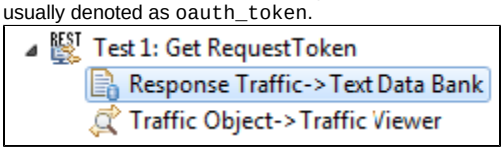
The following example uses the REST Client to send request messages to the server. You can alternatively use the Messaging Client the same manner.

Obtain and Authorize a Request Token from the Service Provider

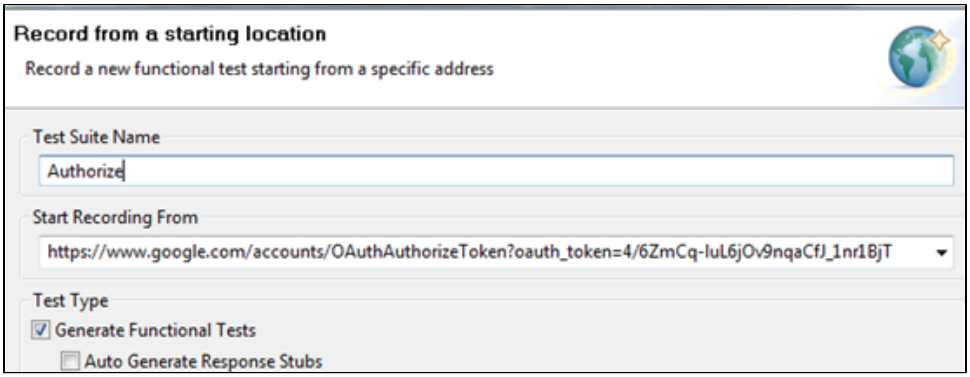
1. Create a new REST Client and configure the settings for the location where the Request Token will be obtained.
2. Click the **HTTP Options** and choose either **HTTP 1.0** or **HTTP 1.1** from the Transport menu.
3. Click the **OAuth Authentication** settings and enable the **Perform Authentication** option. The other fields necessary to complete the OAuth Authentication configuration will become active.
4. Enter the consumer key and consumer secret in the Consumer Key and Consumer Secret fields.
5. Choose **Obtain Request Token** from the OAuth Mode menu.
6. (Optional) Specify a scope in the Scope field.
7. (Optional) Add additional OAuth parameters under OAuth Parameters.



8. Attach a Text Data Bank to the Response Traffic of the REST Client and extract the Request Token and the Request Token Secret. The token is usually denoted as `oauth_token`.

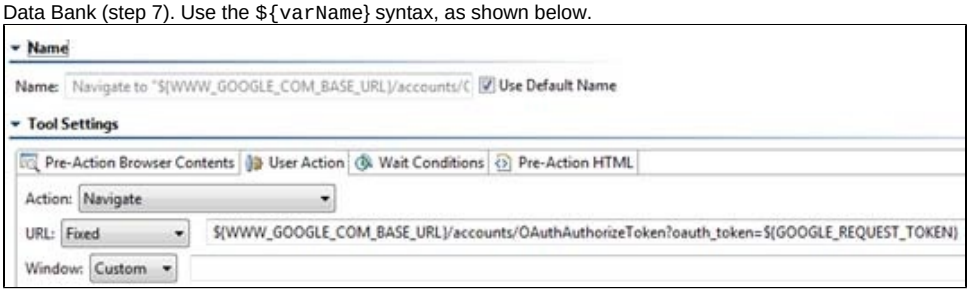


9. Choose **File > New > Test (.tst) File** from the main menu and choose your project.
10. Enter a name for the file and click **Next**.
11. Choose **Web > Record web scenario** and click **Next**.
12. Choose **Record new web scenario** and click **Next**.
13. In the Start Recording From field, enter the URL to obtain the verification code. Add an `oauth_token` parameter and specify the value of the request token obtained in the step 8.



Once the browser launches, it will display the login page of the server that is hosting the protected resource.

14. Sign-in by providing the user's login credentials (Username/Password). Once authorized, the browser will redirect you to a new page with a verification code.
15. After you see the verification code, exit the recording by closing the browser.
16. Attach a Browser Data Bank to the Browser Contents (rendered HTML) and extract the value of the verification code.
17. Open the Browser Playback tool and replace the literal Request Token string with the Request Token data source column generated by the Text Data Bank (step 7). Use the `${varName}` syntax, as shown below.



Exchange the Request Token for an Access Token

1. Create a new REST Client and configure the settings for the location where the Request Token should be exchanged for the Access Token.
2. Click the **HTTP Options** and choose either **HTTP 1.0** or **HTTP 1.1** from the Transport menu.
3. Click the **OAuth Authentication** settings and enable the **Perform Authentication** option. The other fields necessary to complete the OAuth Authentication configuration will become active.
4. Enter the consumer key and consumer secret in the Consumer Key and Consumer Secret fields.
5. Choose **Exchange Request Token for Access Token** from the OAuth Mode menu.
6. Parameterize the Request Token and Request Token Secret fields from the Text Data Bank extractions.
7. Parameterize the Verification Code field from the Browser Data Bank.
8. Attach a Text Data Bank to the Response Traffic of the REST Client and extract the Access Token (usually denoted as oauth_token) and the Access Token Secret (usually denoted as oauth_token_secret).

Sign OAuth Requests to Access Protected Resources

1. Create a new REST Client and configure the settings for the location where the Request Token should be exchanged for the Access Token.
2. Click the **HTTP Options** and choose either **HTTP 1.0** or **HTTP 1.1** from the Transport menu.
3. Click the **OAuth Authentication** settings and enable the **Perform Authentication** option. The other fields necessary to complete the OAuth Authentication configuration will become active.
4. Enter the consumer key and consumer secret in the Consumer Key and Consumer Secret fields.
5. Choose **Sign Request for OAuth Authentication** from the OAuth Mode menu.
6. Parameterize the Access Token and Access Token Secret fields from the Text Data Bank extraction.
7. Request the user's private resources. This should be possible because the Access Token has been obtained.