OAuth Plug-in for Taverna Workbench

Mark Borkum (University of Southampton)
m.i.borkum@soton.ac.uk / @markborkum / github.com/markborkum

"OAuth is an open standard for authorization. It allows users to share their private resources (e.g. photos, videos, contact lists) stored on one site with another site without having to hand out their credentials, typically supplying username and password tokens instead. Each token grants access to a specific site (e.g., a video editing site) for specific resources (e.g., just videos from a specific album) and for a defined duration (e.g., the next 2 hours). This allows a user to grant a third party site access to their information stored with another service provider, without sharing their access permissions or the full extent of their data."

From Wikipedia, the free encyclopaedia

Table of Contents

List of Figures

1 Introduction

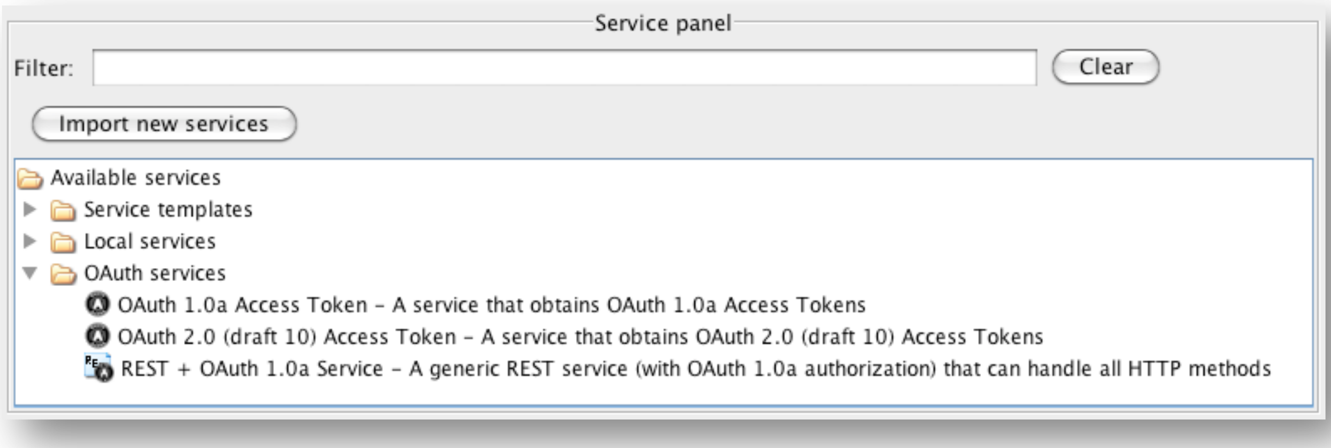
This plug-in enables the use of OAuth-protected REST services in Taverna workflows.

The following versions of the OAuth protocol are supported:

1.0a	http://oauth.net/core/1.0a/
2.0 (draft 10)	http://tools.ietf.org/html/draft-ietf-oauth-v2-10

2 OAuth Services

Three new services are added to the "Service panel" component (shown in Figure 1).



Available services
- Service templates
- Local services
- OAuth services
  - OAuth 1.0a Access Token – A service that obtains OAuth 1.0a Access Tokens
  - OAuth 2.0 (draft 10) Access Token – A service that obtains OAuth 2.0 (draft 10) Access Tokens
  - REST + OAuth 1.0a Service – A generic REST service (with OAuth 1.0a authorization) that can handle all HTTP methods

Figure 1: Taverna Workbench "Service panel" component (with additional "OAuth services" item).

For OAuth 1.0a, two new services are added:

- **OAuth 1.0a Access Token** A service that obtains OAuth 1.0a access tokens.
- **REST + OAuth 1.0a Service** A generic REST service (with OAuth 1.0 authorization) that can handle all HTTP methods.

For OAuth 2.0 (draft 10), one new service is added:

- **OAuth 2.0 (draft 10) Access Token** A service that obtains OAuth 2.0 (draft 10) access tokens.

3 OAuth 1.0a

In Figure 2, we given a depiction of a workflow that obtains an OAuth 1.0a access token and requests a protected resource from Twitter.

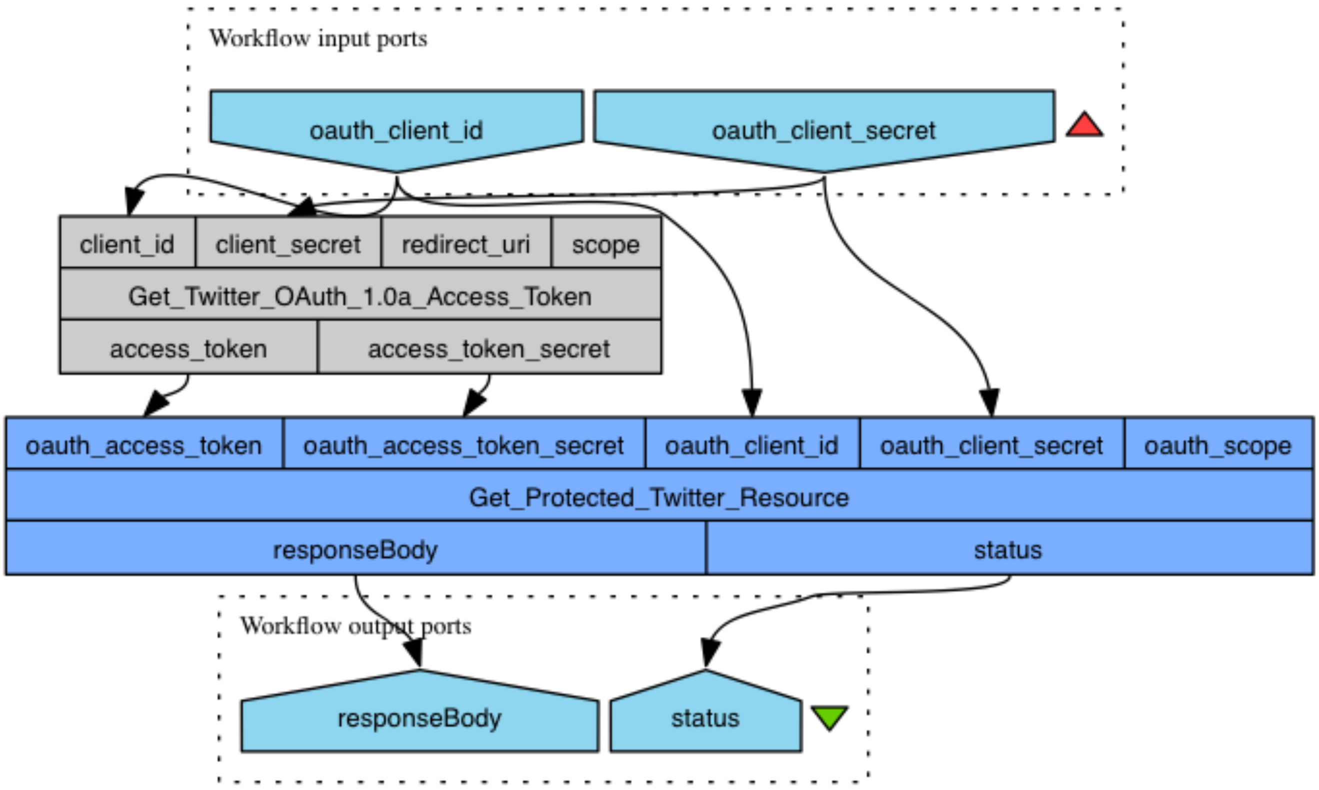


Figure 2: Depiction of workflow that obtains an OAuth 1.0a access token and requests a protected resource from Twitter.

The workflow has two input ports, `oauth_client_id` and `oauth_client_secret`, which correspond to the "consumer key" and "consumer secret" tokens provided by Twitter (see Section 3.1).

Note: *In accordance with version 1.0a of the OAuth protocol, both workflow input ports, and both access token components must be supplied to all subsequent OAuth services.*

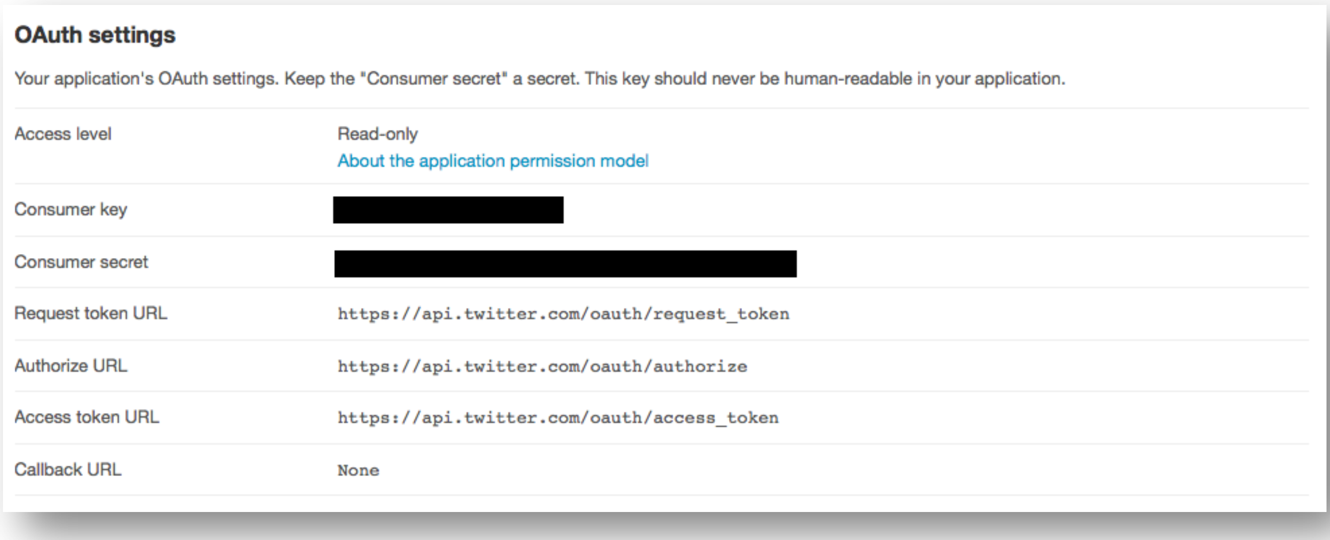
The workflow has two services, `Get_Twitter_OAuth_1.0a_Access_Token` and `Get_Protected_Twitter_Resource`. The first service obtains an OAuth 1.0a access token and access token secret. The second service uses the access token, access token secret, consumer key and consumer secret in order to sign and send an HTTP request for a protected Twitter resource.

Finally, the workflow has two output ports, `responseBody` and `status`, which correspond to the output ports of the second service.

3.1 Registering Taverna Workbench with Twitter

1. Open a Web browser and navigate to https://dev.twitter.com/apps/
2. Click the hyperlink titled "**Create a new application**".
3. Complete the form.
   a. **Name** `Taverna Workbench`
   b. **Description** `Powerful, scalable, open source & domain independent tools for designing and executing workflows. Access to 3500+ resources.`
   c. **Website** `http://www.taverna.org.uk/`
   d. **Callback URL** Leave this field blank.
4. Submit the form.
   a. If the form is submitted successfully, then the Web browser should be redirected to a page that describes the new application.
      i. https://dev.twitter.com/apps/:id/show (where "**:id**" is replaced with the unique identifier for the application).

The application configuration is given in the "OAuth settings" section (shown in Figure 3).



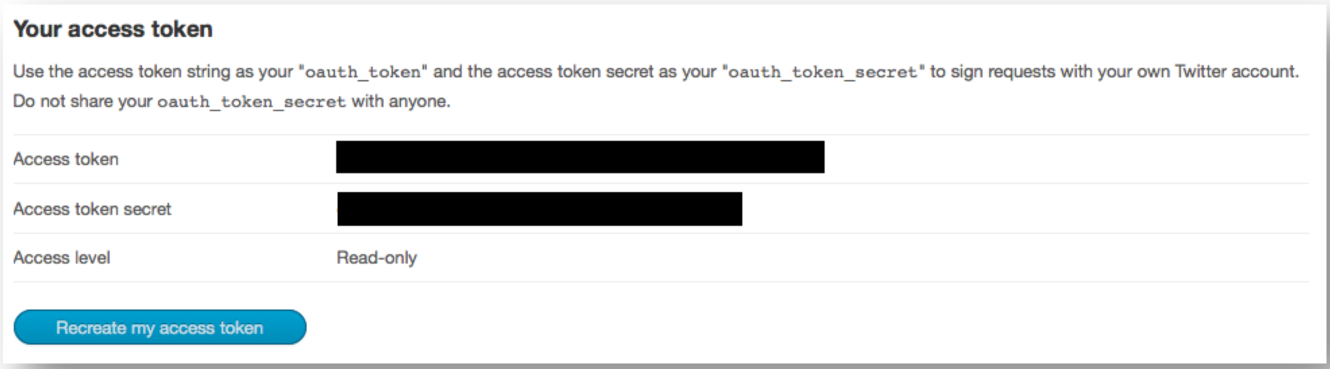
OAuth settings

Your application's OAuth settings. Keep the "Consumer secret" a secret. This key should never be human-readable in your application.

Access level	Read-only About the application permission model
Consumer key	[redacted]
Consumer secret	[redacted]
Request token URL	`https://api.twitter.com/oauth/request_token`
Authorize URL	`https://api.twitter.com/oauth/authorize`
Access token URL	`https://api.twitter.com/oauth/access_token`
Callback URL	None

Figure 3: "OAuth settings" section of Twitter application description page.

A long-term access token is given in the "Your access token" section (shown in Figure 4).



Your access token

Use the access token string as your "`oauth_token`" and the access token secret as your "`oauth_token_secret`" to sign requests with your own Twitter account. Do not share your `oauth_token_secret` with anyone.

Access token	[redacted]
Access token secret	[redacted]
Access level	Read-only

Figure 4: "Your access token" section of Twitter application description page.

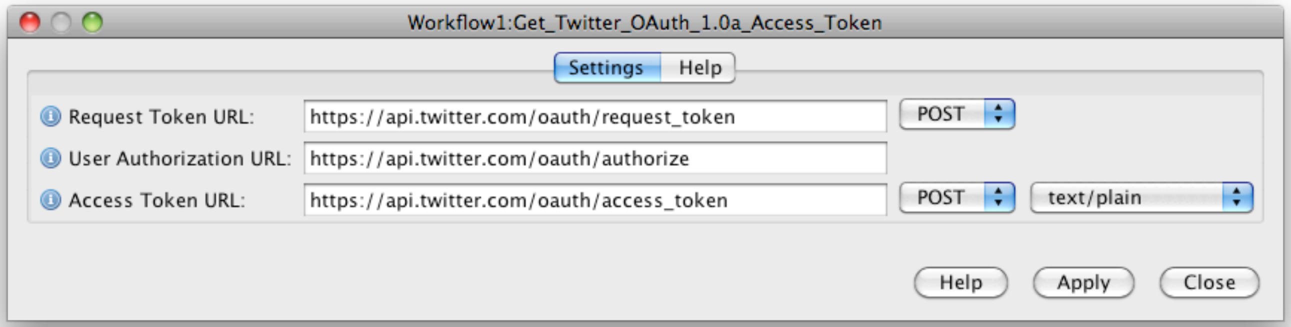
Note: *Clicking the hyperlink titled "Recreate my access token" will refresh the long-term access token.*

3.2 Adding and Configuring an “OAuth 1.0a Access Token” Service

1. Expand **OAuth services** under **Available services** in the **Service panel**.
2. Select **OAuth 1.0a Access Token**.
3. Drag **OAuth 1.0a Access Token** and drop it into the **Workflow diagram**.
4. Right-click on the service in the **Workflow diagram**, and select the “**Configure…**” menu item.
5. In the dialog that appears, configure the **OAuth 1.0a Access Token** as explained next.

The configuration dialog for an “OAuth 1.0a Access Token” service (shown in Figure 5) enables the following parameters to be modified:

- **Request Token**
  - **URL** The URL of the service that provides request tokens.
  - **Method** The method to use for the HTTP request to the service that provides request tokens (GET or POST).
- **User Authorization URL** The URL of the service that provides authorization codes.
- **Access Token**
  - **URL** The URL of the service that provides access tokens.
  - **Method** The method to use for the HTTP request to the service that provides access tokens (GET or POST).
  - **Content Type** The expected content type of the response from the service that provides access tokens (“application/json” or “text/plain”).


Workflow1:Get_Twitter_OAuth_1.0a_Access_Token

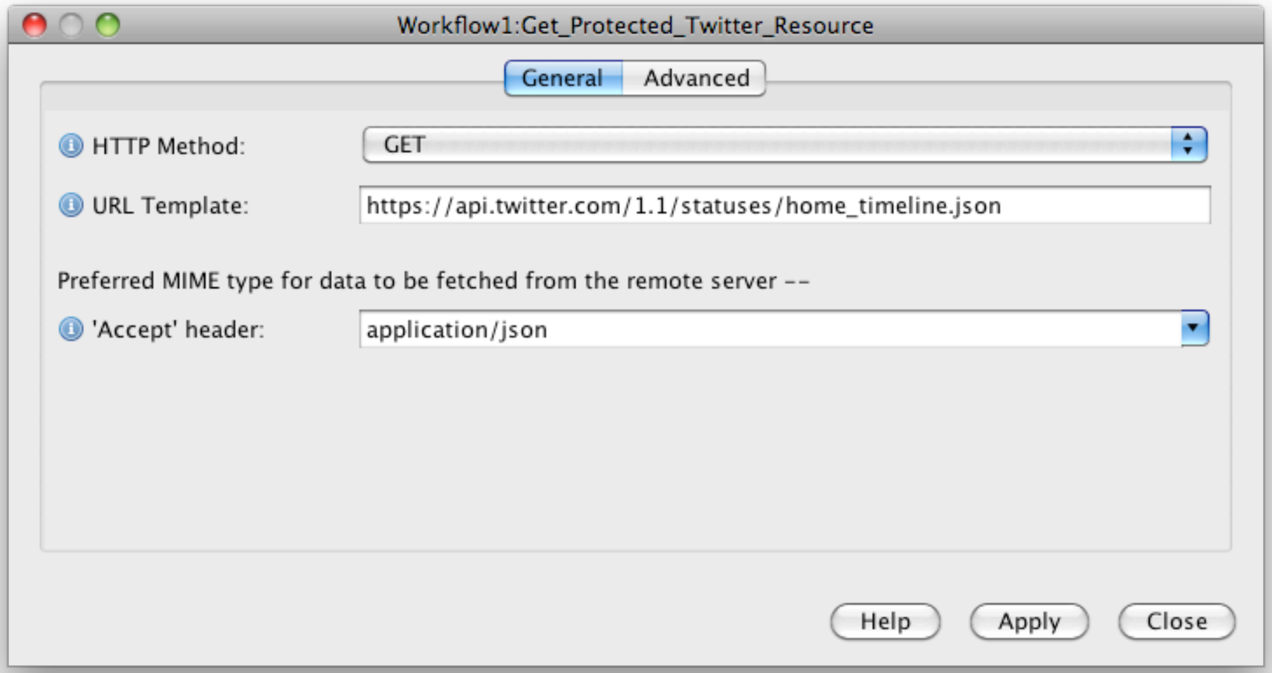
Request Token URL: https://api.twitter.com/oauth/request_token POST
User Authorization URL: https://api.twitter.com/oauth/authorize
Access Token URL: https://api.twitter.com/oauth/access_token POST text/plain

Figure 5: Configuration dialog for "OAuth 1.0a Access Token" service.

3.3 Adding and Configuring a "REST + OAuth 1.0a Service"

1. Expand **OAuth services** under **Available services** in the **Service panel**.
2. Select **REST + OAuth 1.0a Service**.
3. Drag **REST + OAuth 1.0a Service** and drop it into the **Workflow diagram**.
4. Right-click on the service in the **Workflow diagram**, and select the "**Configure...**" menu item.
5. In the dialog that appears, configure the **REST + OAuth 1.0a Service** as explained next.

The configuration dialog for a "REST + OAuth 1.0a Service" (shown in Figure 6) is identical to that of a *normal* REST service.



HTTP Method: GET

URL Template: https://api.twitter.com/1.1/statuses/home_timeline.json

Preferred MIME type for data to be fetched from the remote server --

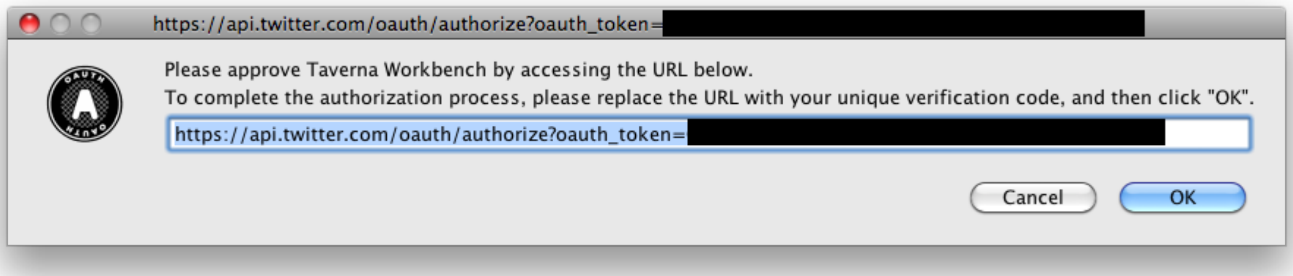
'Accept' header: application/json

Figure 6: Configuration dialog for "REST + OAuth 1.0a Service".

Note: *Each "REST + OAuth 1.0a Service" has five additional input ports, whose names are prefixed with "oauth_" (to avoid conflicts), which are required by the OAuth 1.0a HTTP request-signing algorithm.*

3.4 Executing the “OAuth 1.0a Access Token” Service

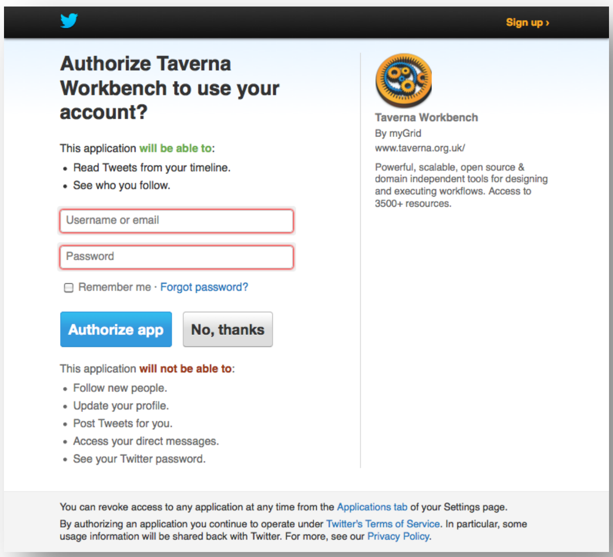
During the execution of an “OAuth 1.0a Access Token” service, the user is prompted to interact with the Taverna Workbench environment in order to perform “out-of-band” (on a separate device, e.g., in a Web browser) user authorization with the OAuth provider. The four stages of the process are depicted in Figure 7. First, a dialog prompts the user to visit a URL in their Web browser. Second, the user authenticates with the OAuth provider. Third, the user receives a verification code. Finally, the user enters the verification code into the dialog, and clicks the “OK” button.


https://api.twitter.com/oauth/authorize?oauth_token=[redacted]

Please approve Taverna Workbench by accessing the URL below.
To complete the authorization process, please replace the URL with your unique verification code, and then click "OK".

https://api.twitter.com/oauth/authorize?oauth_token=[redacted]

(a) User Authorization URL dialog (before).



Authorize Taverna Workbench to use your account?

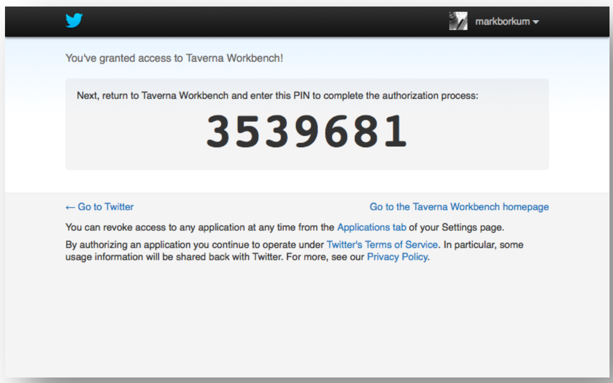
This application will be able to:
- Read Tweets from your timeline.
- See who you follow.

Username or email
Password
Remember me · Forgot password?

This application will not be able to:
- Follow new people.
- Update your profile.
- Post Tweets for you.
- Access your direct messages.
- See your Twitter password.

Taverna Workbench
By myGrid
www.taverna.org.uk/
Powerful, scalable, open source & domain independent tools for designing and executing workflows. Access to 3500+ resources.

You can revoke access to any application at any time from the Applications tab of your Settings page.
By authorizing an application you continue to operate under Twitter's Terms of Service. In particular, some usage information will be shared back with Twitter. For more, see our Privacy Policy.



You've granted access to Taverna Workbench!

Next, return to Taverna Workbench and enter this PIN to complete the authorization process:

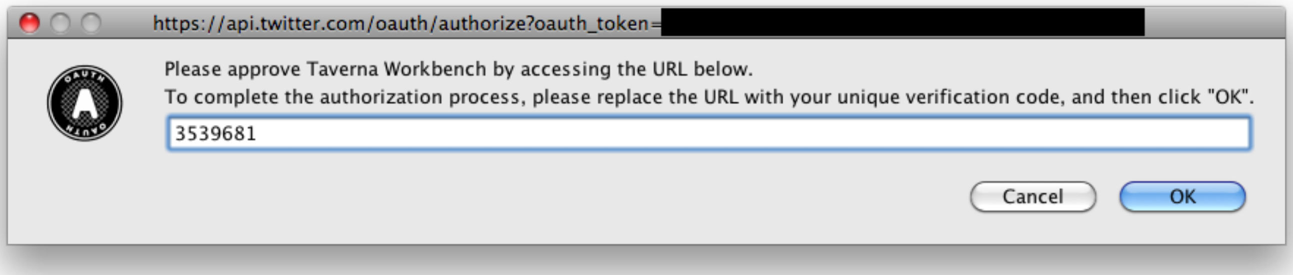
3539681

← Go to Twitter
Go to the Taverna Workbench homepage

You can revoke access to any application at any time from the Applications tab of your Settings page.
By authorizing an application you continue to operate under Twitter's Terms of Service. In particular, some usage information will be shared back with Twitter. For more, see our Privacy Policy.

(c) Screen-shot of Web browser displaying “out-of-band” verification code.

(b) Screen-shot of Web browser displaying User Authorization URL.


https://api.twitter.com/oauth/authorize?oauth_token=[redacted]

Please approve Taverna Workbench by accessing the URL below.
To complete the authorization process, please replace the URL with your unique verification code, and then click "OK".

3539681

(d) User Authorization URL dialog (after).

Figure 7: User interaction process for obtaining an OAuth verification code.

4 OAuth 2.0 (draft 10)

In Figure 8, we give a depiction of a workflow that obtains an OAuth 2.0 (draft 10) access token and requests a protected resource from blog3.

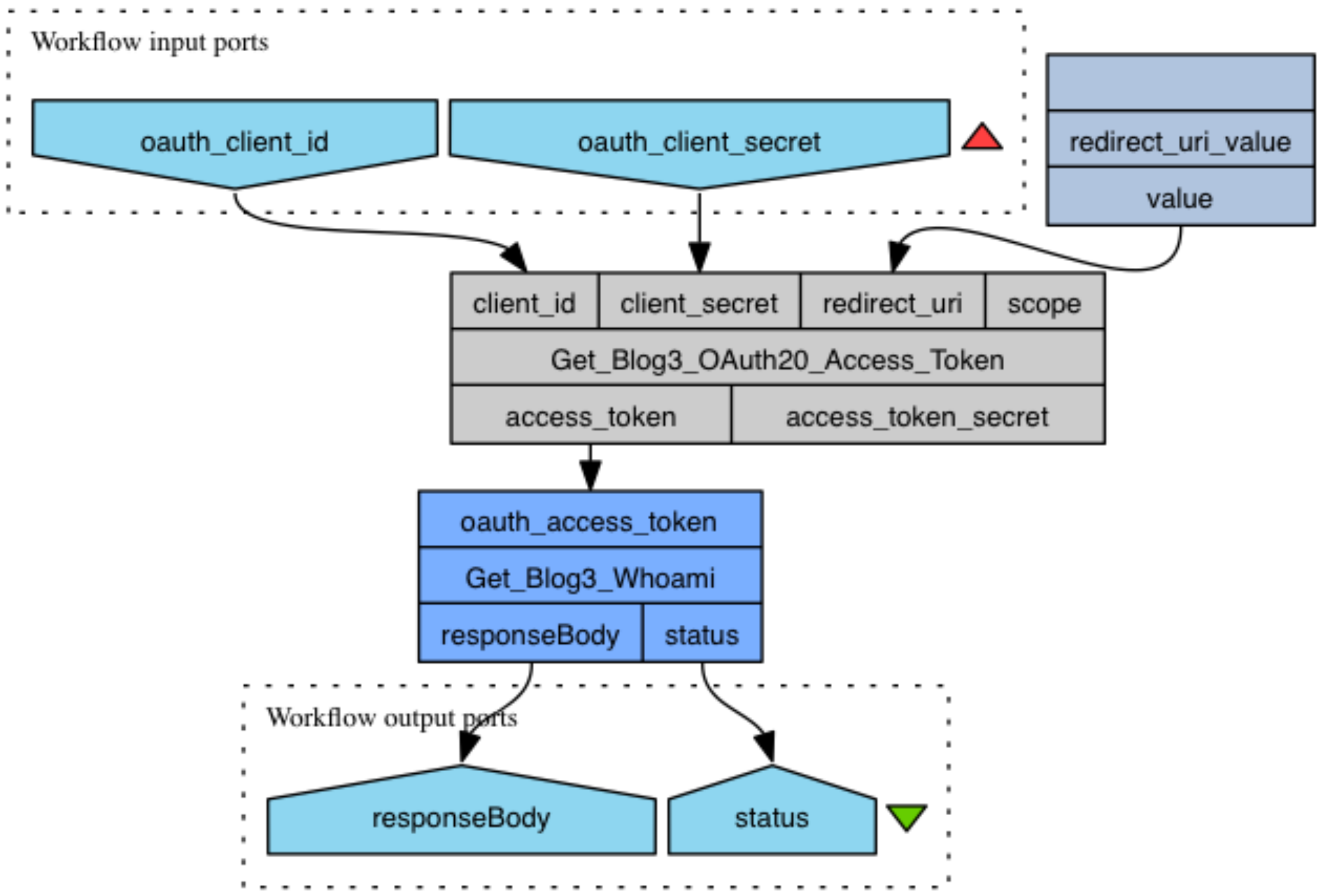


Figure 8: Depiction of workflow that obtains an OAuth 2.0 (draft 10) access token and requests a protected resource from blog3.

The workflow has two input ports, `oauth_client_id` and `oauth_client_secret`, which correspond to the "consumer key" and "consumer secret" tokens provided by blog3 (see Section 4.1).

The workflow has one string constant `redirect_uri_value`, which is set to http://localhost/.

The workflow has two services, `Get_Blog3_OAuth20_Access_Token` and `Get_Blog3_Whoami`. The first service obtains an OAuth 2.0 (draft 10) access token and access token secret. The second service uses the access token in order to sign and send an HTTP request for a protected blog3 resource.

Finally, the workflow has two output ports, `responseBody` and `status`, which correspond to the output ports of the second service.

4.1 Registering Taverna Workbench with blog^3

1. Open a Web browser and navigate to http://blog3-demo.mylabnotebook.ac.uk/my_apps/
2. Click the hyperlink titled "**Register New OAuth Client**".
3. Complete the form.
    a. **Name** `Taverna Workbench`
    b. **Description** `Powerful, scalable, open source & domain independent tools for designing and executing workflows. Access to 3500+ resources.`
    c. **Website** `http://www.taverna.org.uk/`
    d. **Callback URL** `http://localhost/`
4. Submit the form.
    a. If the form is submitted successfully, then the Web browser should be redirected to a page that describes the new application.
        i. http://blog3-demo.mylabnotebook.ac.uk/my_apps/:id (where "**:id**" is replaced with the unique identifier for the application).

The application configuration is given in the "Details" page (shown in Figure 9).

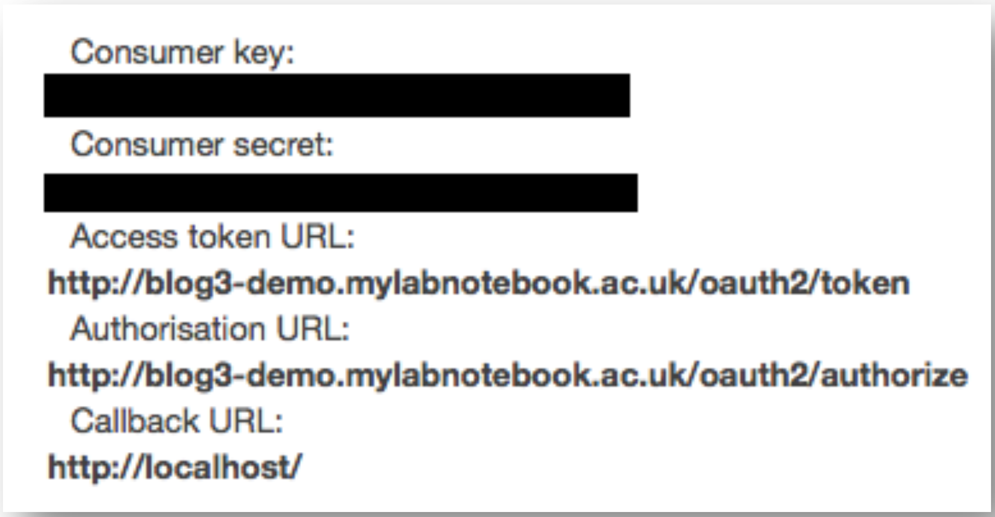

Consumer key:
[redacted]
Consumer secret:
[redacted]
Access token URL:
http://blog3-demo.mylabnotebook.ac.uk/oauth2/token
Authorisation URL:
http://blog3-demo.mylabnotebook.ac.uk/oauth2/authorize
Callback URL:
http://localhost/

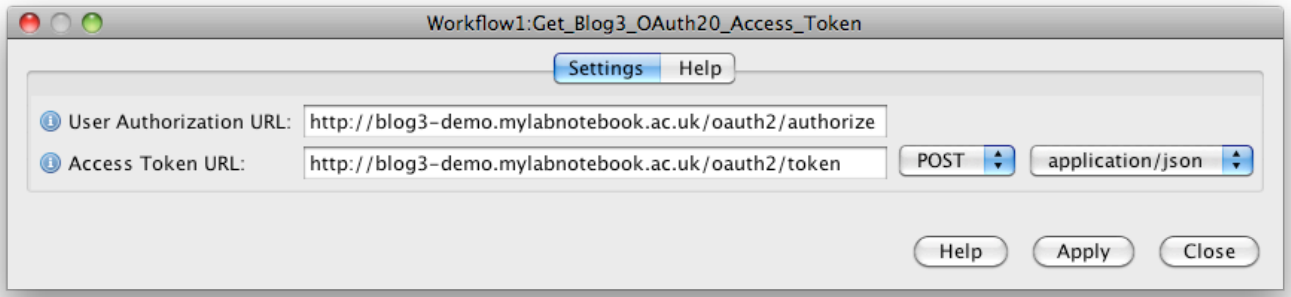
Figure 9: Excerpt of blog^3 OAuth client description page.

4.2 Adding and Configuring "OAuth 2.0 (draft 10) Access Token" Service

1. Expand **OAuth services** under **Available services** in the **Service panel**.
2. Select **OAuth 2.0 (draft 10) Access Token**.
3. Drag **OAuth 2.0 (draft 10) Access Token** and drop it into the **Workflow diagram**.
4. Right-click on the service in the **Workflow diagram**, and select the "**Configure…**" menu item.
5. In the dialog that appears, configure the **OAuth 2.0 (draft 10) Access Token** as explained next.

The configuration dialog for an "OAuth 2.0 (draft 10) Access Token" service (shown in Figure 10) enables the following parameters to be modified:

- **User Authorization URL** The URL of the service that provides authorization codes.
- **Access Token**
  - **URL** The URL of the service that provides access tokens.
  - **Method** The method to use for the HTTP request to the service that provides access tokens (GET or POST).
  - **Content Type** The expected content type of the response from the service that provides access tokens ("application/json" or "text/plain").


Workflow1:Get_Blog3_OAuth20_Access_Token

User Authorization URL: http://blog3-demo.mylabnotebook.ac.uk/oauth2/authorize

Access Token URL: http://blog3-demo.mylabnotebook.ac.uk/oauth2/token POST application/json

Figure 10: Configuration dialog for "OAuth 2.0 (draft 10) Access Token" service.

4.3 Adding and Configuring "REST + OAuth 2.0 (draft 10) Service"

In contrast to version 1.0a, version 2.0 of the OAuth protocol does not specify a default HTTP request-signing algorithm. For this reason, a specialised "REST + OAuth 2.0 (draft 10) Service" is not provided. Instead, users are required to add and configure a *normal* "REST Service".

1. Expand **Service templates** under **Available services** in the **Service panel**.
2. Select **REST Service**.
3. Drag **REST Service** and drop it into the **Workflow diagram**.
4. In the dialog that appears, configure the **REST Service** as explained next.

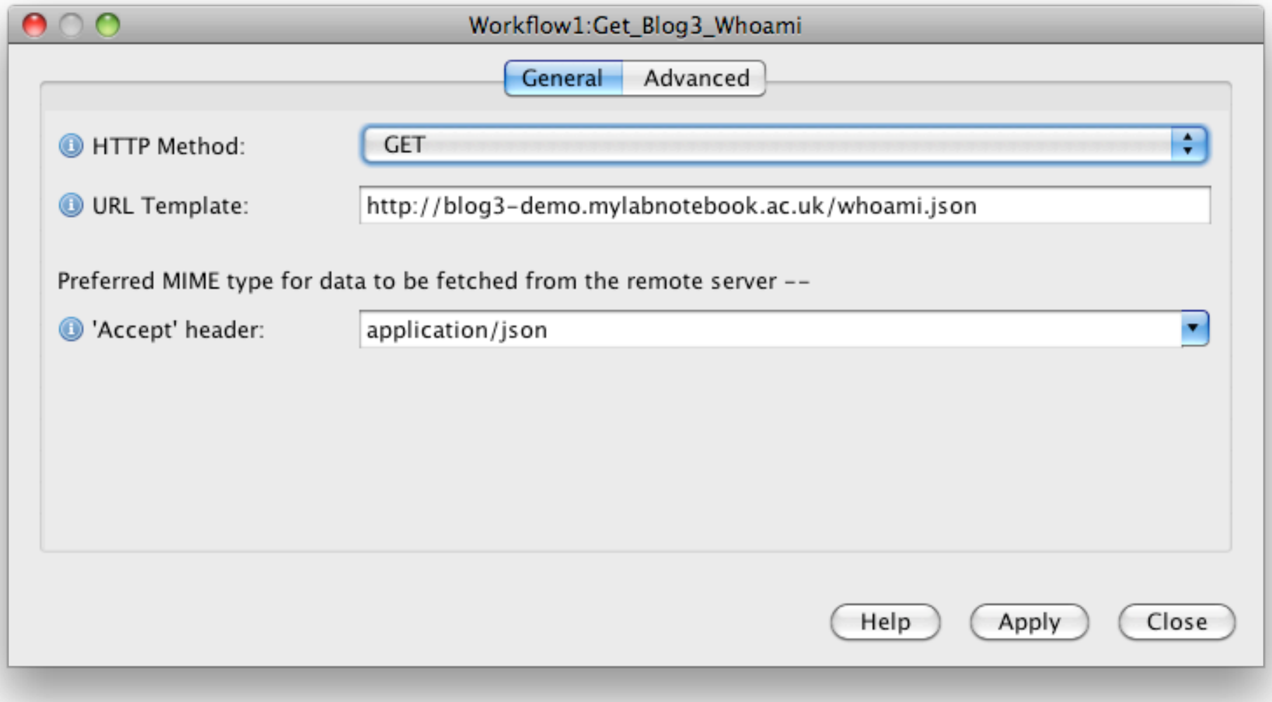
4.3.1 Advanced Configuration of "REST Service" for blog^3

In Figures 11 and 12, we give screen-shots of the "General" and "Advanced" tabs of the configuration dialog for a "REST Service" that requests a protected resource from blog^3.

To sign an HTTP request for blog^3, a new header is added to the request:

- **Name** `Authorization`
- **Value** `Bearer {oauth_access_token}`

The URL signature handler interprets the value of the header as a template, and an `oauth_access_token` input port is automatically created.


Workflow1:Get_Blog3_Whoami

General | Advanced

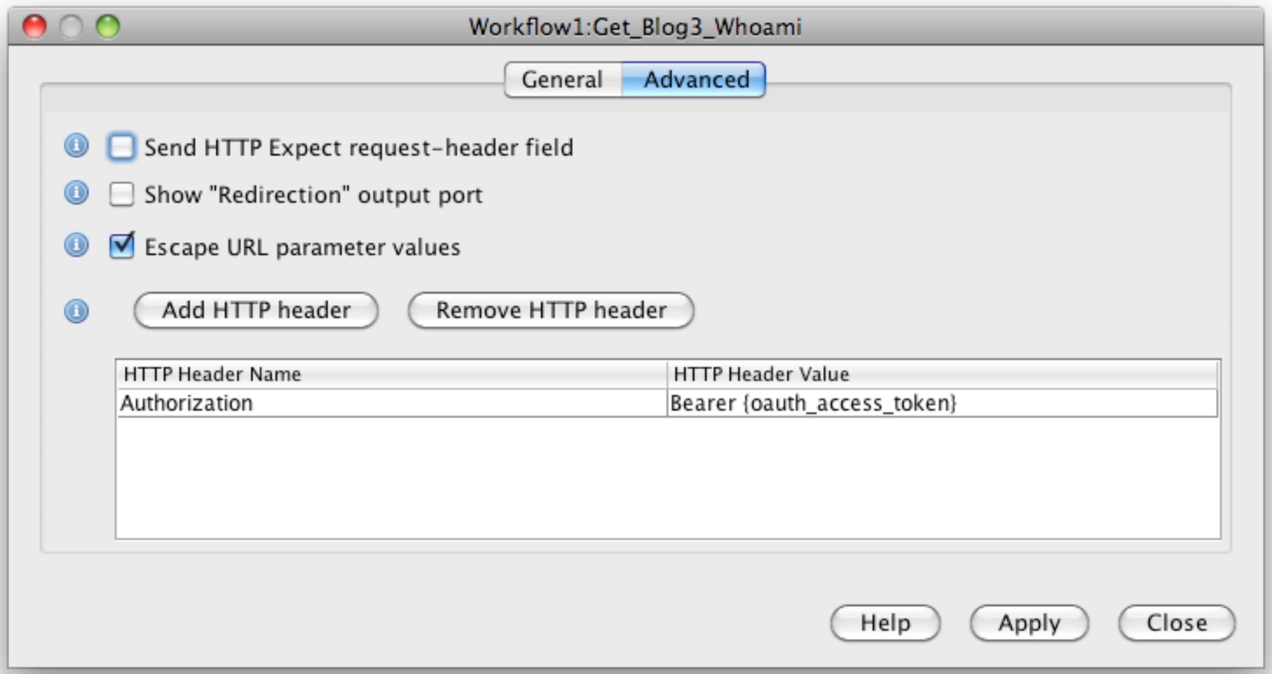
HTTP Method: GET

URL Template: http://blog3-demo.mylabnotebook.ac.uk/whoami.json

Preferred MIME type for data to be fetched from the remote server --

'Accept' header: application/json

Figure 11: "General" tab of configuration dialog for "REST Service".


Workflow1:Get_Blog3_Whoami

General | Advanced

- [ ] Send HTTP Expect request-header field
- [ ] Show "Redirection" output port
- [x] Escape URL parameter values

HTTP Header Name	HTTP Header Value
Authorization	Bearer {oauth_access_token}

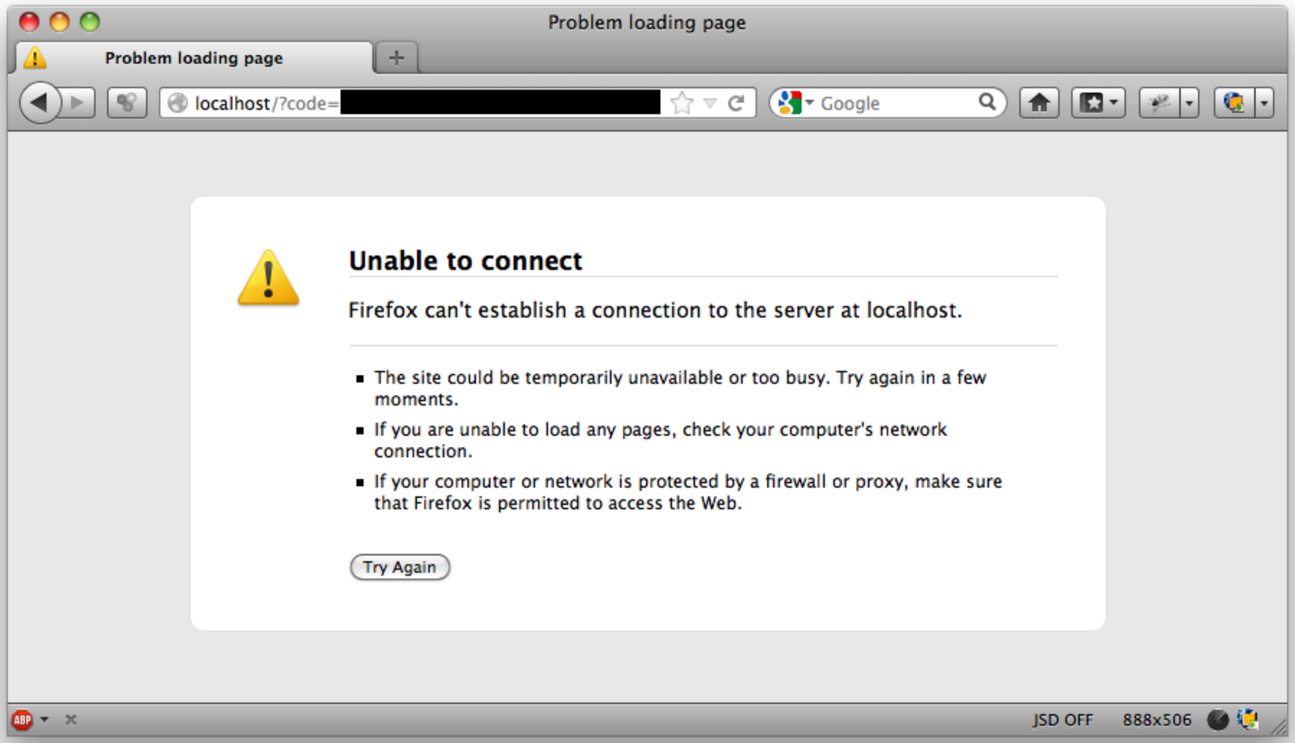
Figure 12: "Advanced" tab of configuration dialog for "REST Service".

4.4 Executing the “OAuth 2.0 (draft 10) Access Token” Service

The process of obtaining an OAuth verification code in version 2.0 (draft 10) of the protocol is identical to that of version 1.0a (described in Section 3.4).

4.4.1 Obtaining an OAuth Verification Code with “localhost” Redirection

In Figure 13, we give a screen-shot of a Web browser showing the result of successfully authenticating with an OAuth provider when http://localhost/ is specified as the “redirect” (or “callback”) URL.



Problem loading page

localhost/?code=[redacted]

Unable to connect

Firefox can't establish a connection to the server at localhost.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

Figure 13: Screen-shot of Web browser showing the result of successfully authenticating with an OAuth provider when http://localhost/ is specified as the “redirect” (or “callback”) URL.

In the figure, the verification code has been included in the URL as the value of the `code` query parameter.