# Mule (Secure Token Service)STS to apply Oauth 2.0 Example Application

To create an OAuth2 provider you must have the **Enterprise Security module** installed in your Anypoint Studio.

Tweetbook application – OAuth service provider(Web service)

* Readers’s social network
* User can make a account & keep a list of books they read(manage virtual library)
* Share the library items
* Post comments & reviews

BookStore app- OAuth service consumer(Web Service)

* Online retailer of books

**User Story:**

1)End users who wish to purchase books and who are Tweetbook users can choose to create a Bookstore account using their Tweetbook account credentials.

2)When a user decide to use their Tweetbook account to create an account with Bookstore, Tweetbook shares the user’s virtual library with Bookstore. Users can then consult their Tweetbook library before purchasing new books.

**Scope:**

Tweetbook has offered its API to Bookstore and is prepared to provide **read-only access to users’ profiles and virtual bookshelves.**

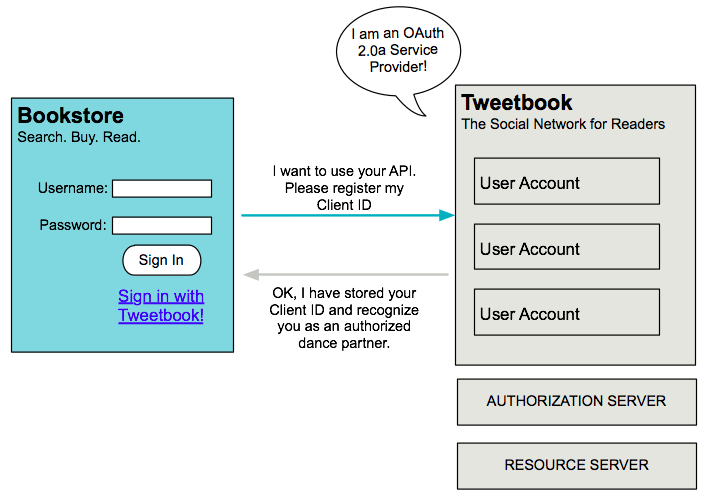
The table below describes the OAuth entities within the context of this example.

| **Entity** | **Description** |
| --- | --- |
| Access Credentials | Login credentials to Tweetbook |
| Resource Owner | The end user |
| Protected Resource | The end user’s account with Tweetbook |
| Service Provider | Tweetbook |
| Consumer | Bookstore |
| Token | Token |
| Scope | Read-only access to the end user’s profile and list of books in the virtual library |

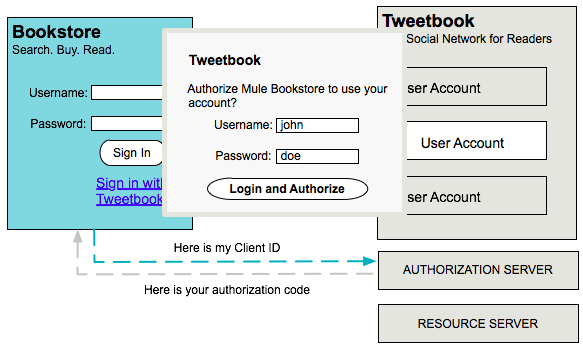
### The OAuth Dance

**Requirements:**

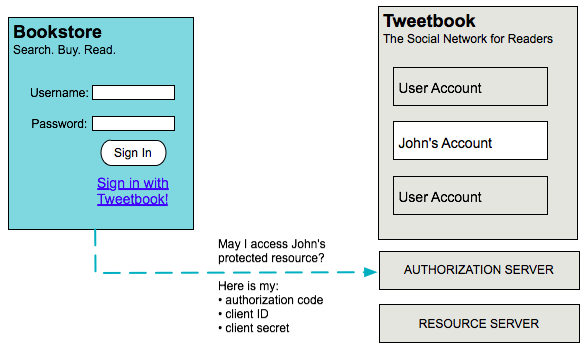
1. Service provider must **define itself as an OAuth Service Provider**
2. Consumer must **register its Client ID with the provider**.(The service provider stores the Ids of registered consumers in its object store)
3. On the Bookstore’s website, the end user — a Tweetbook account holder — decides to create a new account using his Tweetbook account credentials. The user clicks on the “Sign in with Tweetbook!” link that Bookstore makes available for this purpose.



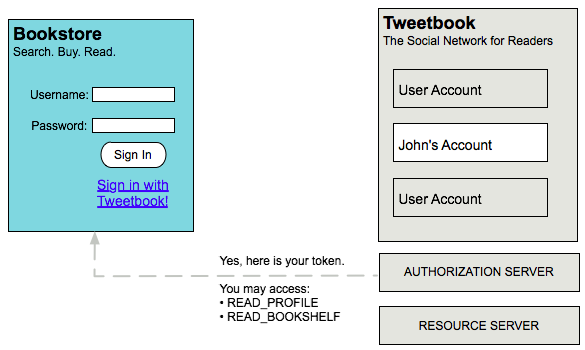
1. Tweetbook and Bookstore use the [Authorization Grant Type](https://docs.mulesoft.com/mule-runtime/3.8/mule-sts-oauth-2.0-example-application#authorization-grant-type) to perform the OAuth dance. The Bookstore directs the user to a URL to log in to his Tweetbook account. In this step, Bookstore calls Tweetbook’s authorization server and provides its client ID (so Tweetbook knows which consumer is asking to dance). The user enters his Tweetbook login credentials. When he clicks **Login and Authorize**, he not only accesses his Tweetbook account, he also authorizes Tweetbook to share read-only access of his profile and bookshelf with Bookstore. Tweetbook sends Bookstore an authorization code.



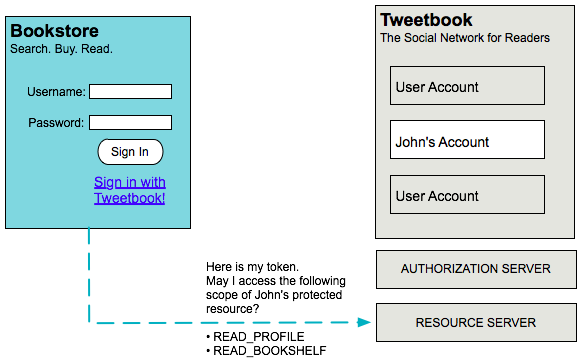
1. Bookstore calls Tweetbook’s authorization server again to request permission to access the protected resource (the user’s Tweetbook account). Bookstore provides its authorization code, client ID and client secret.



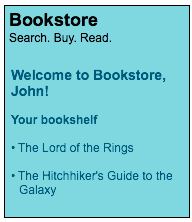
1. Tweetbook grants permission to access the protected resource by providing a token. Within the token, it defines the scope (that is, specific data) which Bookstore may access. In this case, the scope is read-only access to the user’s profile and virtual bookshelf.



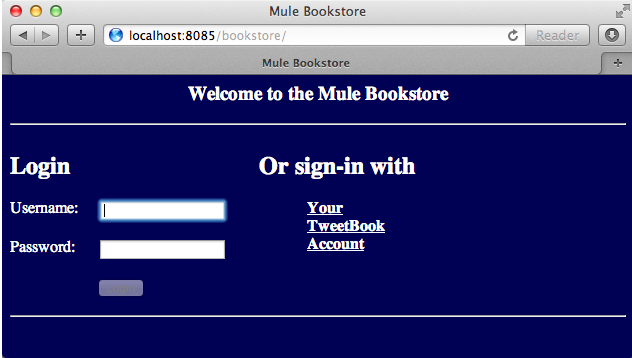
1. Bookstore calls Tweetbook’s resource server to request the user’s protected resource. It offers the token Tweetbook provided as both proof of its identity and its authorization to acquire the resource.

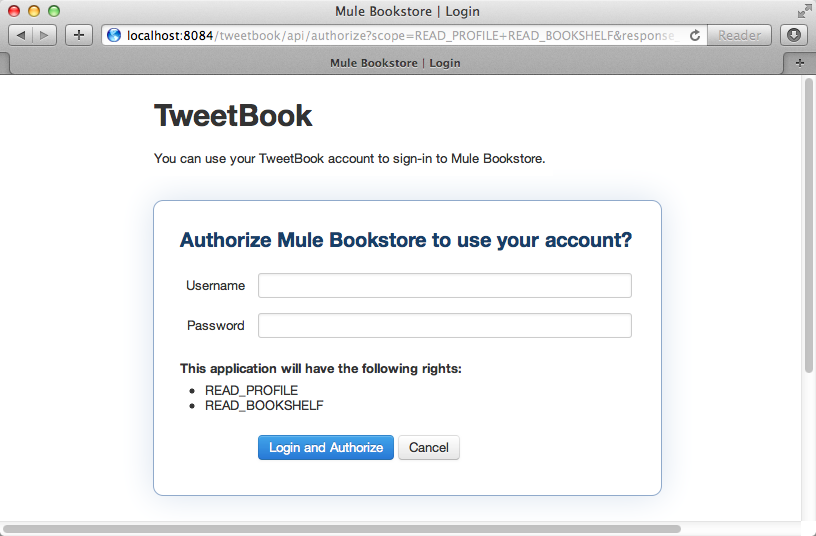


1. Bookstore allows the user to enter his new account where his virtual bookshelf is pre-populated with the list of books in his Tweetbook account.



<http://localhost:8085/bookstore>



1. Click the **Your Tweetbook Account** link to sign in with Tweetbook.
2. Bookstore opens a Tweetbook login in window.
3. 
4. Tweetbook grants the Bookstore application access to your profile and bookshelf data. Bookstore imports the data, then logs you into your new user account on Bookstore. Then, Bookstore displays the contents of your Tweetbook bookshelf.
5. 

### Authorization Grant Type

There are four ways that a consumer can obtain authorization to dance with an OAuth service provider.

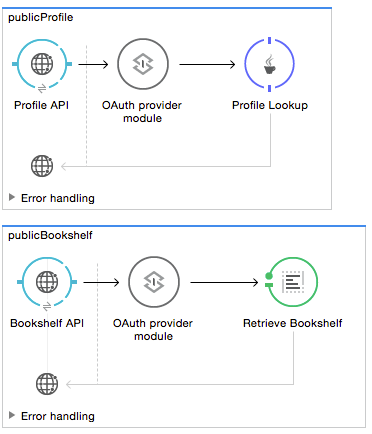
1. Authorization Code
2. Implicit
3. Resource Owner Password Credentials
4. Client Credentials

This example application uses the Authorization Code grant type which, as described above, involves a multi-step process to authorize sharing of a protected resource. For a service provider, this grant type involves the use of both an authorization server (responsible for confirming and granting permission to access the protected resource) and a resource server (responsible for providing access to the protected resource). A consumer must ask for a service provider’s permission to ask for protected resources — like a suitor asking a chaperone’s permission to ask a woman to dance. If the chaperone (authorization server) grants permission to ask, the suitor (consumer) may then ask the woman (resource server) to dance.

Refer to [Authorization Grant Types](https://docs.mulesoft.com/mule-runtime/3.8/authorization-grant-types) for more information on the other types of authorization grants.

## Oauth 2.0 Configuration in Tweetbook

The Tweetbook application uses the Mule Secure Token Service to apply Oauth 2.0 to the API it exposes. By including the OAuth2 message processor in the Web service’s flow, Tweetbook defines itself as an OAuth2 service provider and enables Mule to perform the OAuth dance with consumers.



### View the Oauth 2.0 Configuration XML

<?xml version="1.0" encoding="UTF-8"?>

<mule xmlns:http="http://www.mulesoft.org/schema/mule/http"

xmlns="http://www.mulesoft.org/schema/mule/core" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:oauth2-provider="http://www.mulesoft.org/schema/mule/oauth2-provider"

xmlns:spring="http://www.springframework.org/schema/beans"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:mule-ss="http://www.mulesoft.org/schema/mule/spring-security"

xmlns:ss="http://www.springframework.org/schema/security"

xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"

xsi:schemaLocation="http://www.mulesoft.org/schema/mule/http http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd

http://www.mulesoft.org/schema/mule/oauth2-provider http://www.mulesoft.org/schema/mule/oauth2-provider/current/mule-oauth2-provider.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-current.xsd

http://www.mulesoft.org/schema/mule/core http://www.mulesoft.org/schema/mule/core/current/mule.xsd

http://www.mulesoft.org/schema/mule/spring-security http://www.mulesoft.org/schema/mule/spring-security/current/mule-spring-security.xsd

http://www.springframework.org/schema/security http://www.springframework.org/schema/security/spring-security.xsd

http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-current.xsd">

<context:property-placeholder location="bookstore.properties" />

<spring:beans>

<ss:authentication-manager id="resourceOwnerAuthenticationManager">

<ss:authentication-provider>

<ss:user-service id="resourceOwnerUserService">

<ss:user name="john" password="doe" authorities="RESOURCE\_OWNER" />

</ss:user-service>

</ss:authentication-provider>

</ss:authentication-manager>

</spring:beans>

<mule-ss:security-manager>

<mule-ss:delegate-security-provider

name="resourceOwnerSecurityProvider"

delegate-ref="resourceOwnerAuthenticationManager" />

</mule-ss:security-manager>

<oauth2-provider:config

name="oauth2Provider"

providerName="TweetBook"

port="8084"

authorizationEndpointPath="tweetbook/api/authorize"

accessTokenEndpointPath="tweetbook/api/token"

resourceOwnerSecurityProvider-ref="resourceOwnerSecurityProvider"

scopes="READ\_PROFILE WRITE\_PROFILE READ\_BOOKSHELF WRITE\_BOOKSHELF" doc:name="OAuth provider module">

<oauth2-provider:clients>

<oauth2-provider:client clientId="${client\_id}" secret="${client\_secret}"

type="CONFIDENTIAL" clientName="Mule Bookstore" description="Mule-powered On-line Bookstore">

<oauth2-provider:redirect-uris>

<oauth2-provider:redirect-uri>http://localhost\*</oauth2-provider:redirect-uri>

</oauth2-provider:redirect-uris>

<oauth2-provider:authorized-grant-types>

<oauth2-provider:authorized-grant-type>AUTHORIZATION\_CODE</oauth2-provider:authorized-grant-type>

</oauth2-provider:authorized-grant-types>

<oauth2-provider:scopes>

<oauth2-provider:scope>READ\_PROFILE</oauth2-provider:scope>

<oauth2-provider:scope>READ\_BOOKSHELF</oauth2-provider:scope>

<oauth2-provider:scope>WRITE\_BOOKSHELF</oauth2-provider:scope>

<oauth2-provider:scope>WRITE\_PROFILE</oauth2-provider:scope>

</oauth2-provider:scopes>

</oauth2-provider:client>

</oauth2-provider:clients>

</oauth2-provider:config>

<http:listener-config name="HTTP\_Listener\_Configuration" host="localhost" port="8084" doc:name="HTTP Listener Configuration" />

<flow name="publicProfile">

<http:listener config-ref="HTTP\_Listener\_Configuration" path="/tweetbook/api/profile" allowedMethods="GET" doc:name="Profile API" />

<oauth2-provider:validate scopes="READ\_PROFILE" config-ref="oauth2Provider" doc:name="OAuth provider module"/>

<component class="org.mule.security.examples.oauth2.ProfileLookupComponent" doc:name="Profile Lookup"/>

</flow>

<flow name="publicBookshelf">

<http:listener config-ref="HTTP\_Listener\_Configuration" path="/tweetbook/api/bookshelf" allowedMethods="GET" doc:name="Bookshelf API"/>

<oauth2-provider:validate scopes="READ\_BOOKSHELF" config-ref="oauth2Provider" doc:name="OAuth provider module"/>

<set-payload value="The Lord of the Rings,The Hitchhiker's Guide to the Galaxy" doc:name="Retrieve Bookshelf"/>

</flow>

</mule>