OSLC and Jazz service access using a browser

Michael Fiedler
IBM Rational and Eclipse Lyo
21 August 2012

Browser Plugins for REST services

- Many plugins available to invoke REST services from browsers
- Some I have used:
  - Firefox
    - RESTClient
  - Chrome
    - Advanced REST Client
    - Postman

OSLC vs Jazz service access

- OSLC
  - RESTful web services for lifecycle integrations
  - Catalog, Service Provider and domain-specific resources (Change Requests, Requirements, Test Cases, etc)
  - CRUD operations via HTTP methods (GET, POST, PUT, DELETE)
  - Some standard headers
    - Accept = application/rdf+xml or application/json
    - Content-Type = application/rdf+xml or application/json
    - OSLC-Core-Version = 2.0
  - Authentication can be OAuth, Form, Basic or other
- IBM Rational Jazz Servers
  - OSLC + additional protocols and patterns
  - Authentication usually Form for users and OAuth for applications
  - “rootservices” protocol for initial service discovery

Scenarios

- Service Discovery
- Querying, Creating and Updating resources
- Jazz rootservices
- The OAuth dance

Service Discovery and Navigation

- The OSLC Service Provider Catalog
  - Listing of service providers & URLs to reach them
  - On Rational Jazz servers
    - Catalog per product (RTC, RQM, RRC)
    - Service provider per project area
- The OSLC Service Provider Document
  - URLs to use for:
    - Querying for resources
    - Creating resources
    - OSLC delegated dialogs (selection and creation)

CRUD operations on artifacts

- Querying or getting resources
  - QueryCapability in service provider = URL for queries
  - Sample query
  - OSLC Paging
    - References to current, previous, next page
- Creating resources
  - CreationFactory in service provider = URL for creation
- Updating resources
  - GET it, then PUT it back
  - Use ETag and If-Match headers to avoid simultaneous update issue
- Deleting resources
  - Use DELETE on the resource URL

Rational Jazz servers

- Rootservices document: enables clients to discover Jazz services
  - OSLC Service Provider Catalog URLs
    - oslc_cm:cmServiceProviders (RTC and RQM)
    - oslc_qm:qmServiceProviders (RQM)
    - oslc_rm:rmServiceProviders (RRC, DOORS too)
  - OAuth-related URLs
    - oauthAccessTokenUrl
    - oauthUserAuthorizationUrl
    - oauthRequestTokenUrl
    - Also:
      - oauthRequestConsumerKeyUrl
      - oauthApprovalModuleUrl

OSLC and Authentication

- Goal: authenticate our application with an OSLC provider
  - Basic Auth
    - Simple to do programmatically
    - Not very secure – should only be done over https
  - FORM Auth
    - More secure, but still should only be done over https
    - Tricky to do programmatically – distinct protocols
    - Library available soon in Eclipse Lyo to help with Rational Jazz servers
  - OAuth 1.0 and 1.0a
    - Even more secure – no need for client program to know user passwords
    - Generally requires user interaction, through. A user must authorize a program's access to protected resources
    - The "OAuth dance", where access is granted to a program, can be frustrating to learn.

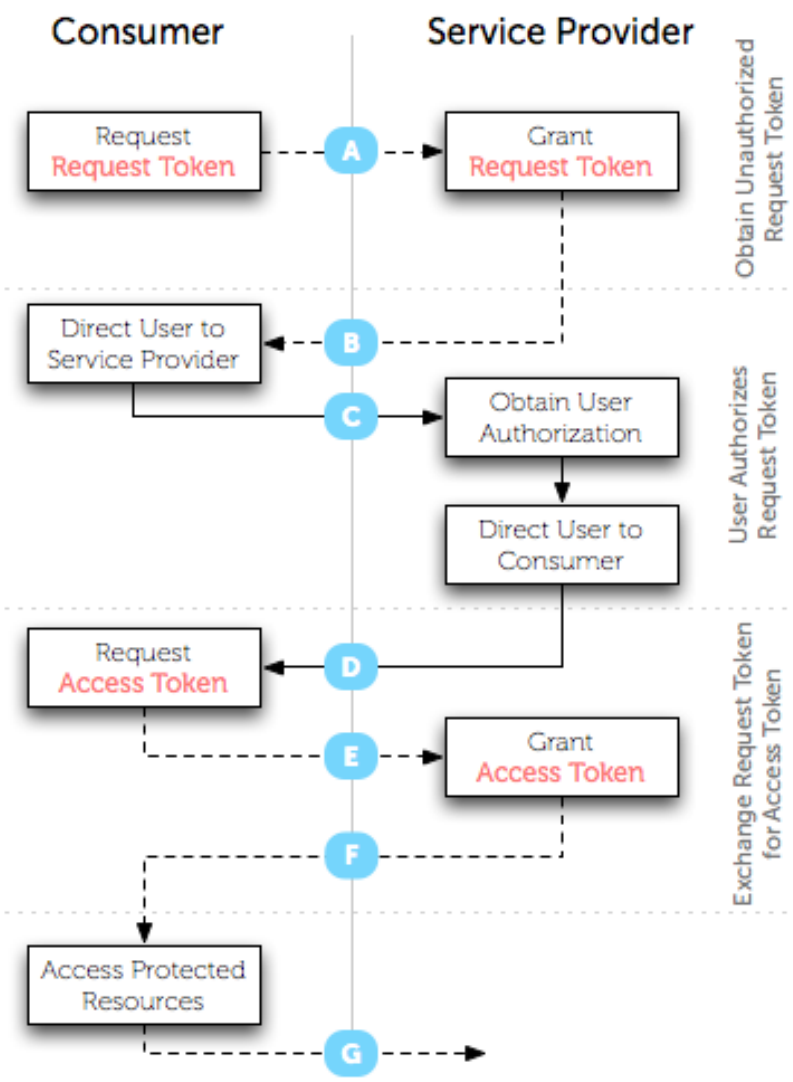
The OAuth Dance

1. Get a consumer key for your application from the provider
   - For Rational Jazz servers, can be manual or programmatic
     - Manual: Admin page of the application -> Consumers page
     - Programmatic: See: https://jazz.net/wiki/bin/view/Main/RootServicesSpecAddendum2
2. Use your consumer key and shared secret to get a temporary request token
3. The user of your application authorizes access to protected resources using the request token
4. Exchange the request token for an access token
5. Access protected resources using the access token

OAUTH AUTHENTICATION FLOW v1.0a

Person Using Web Browser / Manual Entry

Consumer / Service Provider



A Consumer Requests Request Token

Request includes
oauth_consumer_key
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)
oauth_callback

B Service Provider Grants Request Token

Response includes
oauth_token
oauth_token_secret
oauth_callback_confirmed

C Consumer Directs User to Service Provider

Request includes
oauth_token (optional)

D Service Provider Directs User to Consumer

Request includes
oauth_token
oauth_verifier

E Consumer Requests Access Token

Request includes
oauth_consumer_key
oauth_token
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)
oauth_verifier

F Service Provider Grants Access Token

Response includes
oauth_token
oauth_token_secret

G Consumer Accesses Protected Resources

Request includes
oauth_consumer_key
oauth_token
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)

Oauth 1.0 vs 1.0a

- OAuth 1.0a addresses a vulnerability in 1.0 known as a session fixation attack
  - Not a purely technical or cryptographic attack. Requires some social engineering to get user to visit the authorization URL using the attackers request token.
  - See: http://hueniverse.com/2009/04/explaining-the-oauth-session-fixation-attack/
- oauth_callback URL is optional in 1.0. Required in 1.0a.
  - When user is redirected back to the application callback URL, an oslc_verifier token will be provided which must be used to exchange request token for access token.
- Rational Jazz products
  - 2.x and 3.0.x support OAuth 1.0
  - 4.0.x support OAuth 1.0 and 1.0a

Resources

- Eclipse Lyo
  - OSLC and Jazz client library
  - Under development
  - Available in http://git.eclipse.org/gitroot/lyo/org.eclipse.lyo.client.git
- OAuth libraries
  - http://oauth.net/code/
- Netflix OAuth test tool
  - http://developer.netflix.com/resources/OAuthTest
- URL Encode/Decode tool
  - http://urldecoder.net/
- RDF Libraries
  - http://www.w3.org/wiki/SemanticWebTools
- Jazz rootservices info
  - https://jazz.net/wiki/bin/view/Main/RootServicesSpec