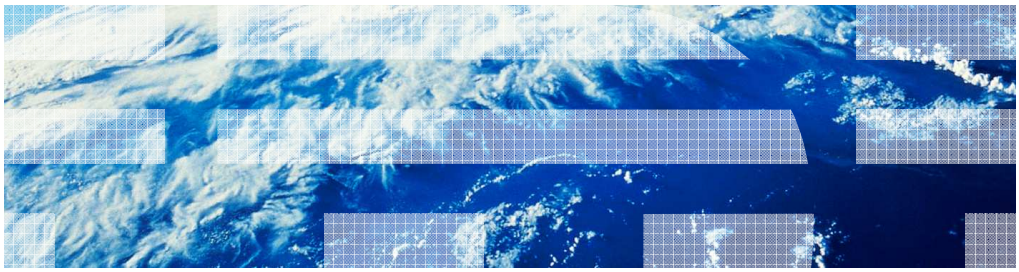
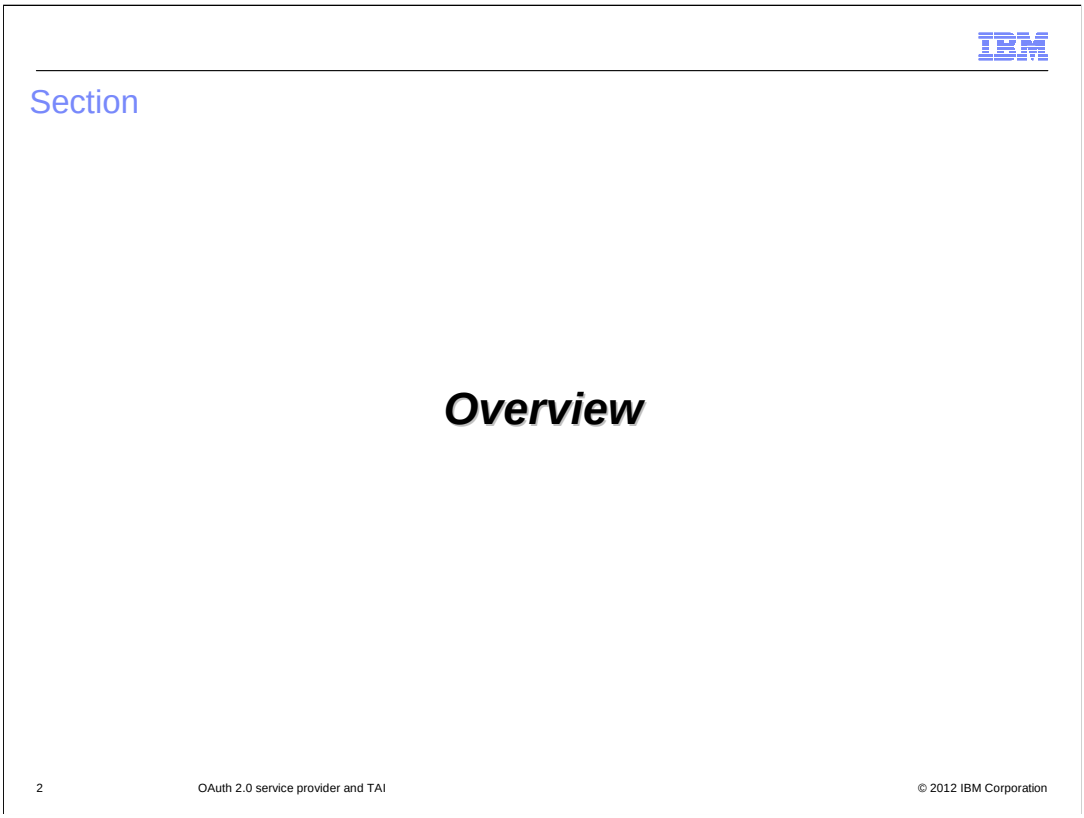
IBM WebSphere Application Server

OAuth 2.0 service provider and TAI



This presentation describes support for OAuth 2.0 included in IBM WebSphere Application Server V7.0.0.25.



This feature provides an OAuth 2.0 service provider to handle all OAuth protocol messages. It also provides a trusted association interceptor (TAI) to validate OAuth 2.0 token when client applications request web resources.

What is OAuth?

- Protocol for delegated authorization.
- Enables you to grant third-party application access to your web resources without sharing passwords

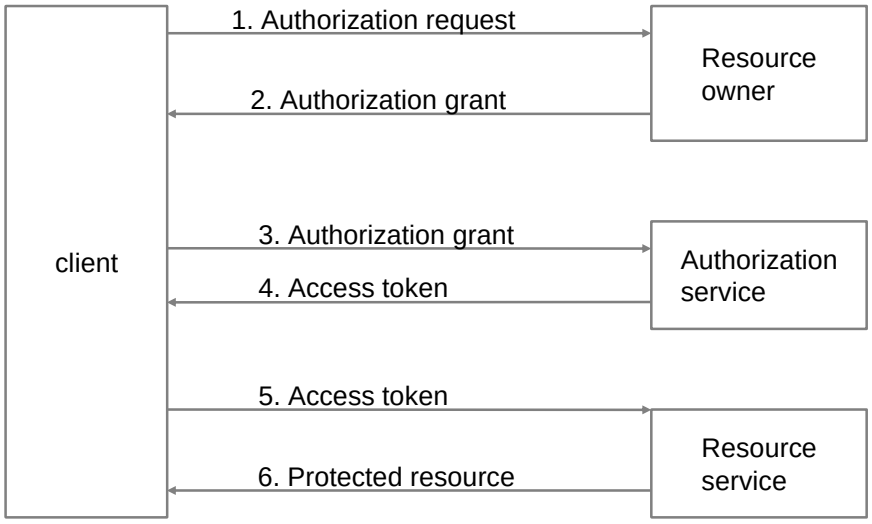
OAuth is a standard to enable a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf.

OAuth versions and specs

- Versions: As a protocol, OAuth has two versions: OAuth 1.0, and OAuth 2.0.
- OAuth 2.0 replaces OAuth 1.0.
- OAuth 2.0 is not back compatible with OAuth 1.0

OAuth 2.0 is an enhancement to the previous OAuth 1.0 specifications, but is not compatible with earlier versions.

Roles and flows in OAuth 2.0



OAuth 2.0 defines four roles: resource owner, resource server, client and authorization server. A resource owner is an entity that is capable of granting access to a protected resource (example, end-user). A resource server is the server hosting the resources protected by access tokens. A client is an application making protected resource requests on behalf of the resource owner, with it's authorization. An authorization server is the server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization.

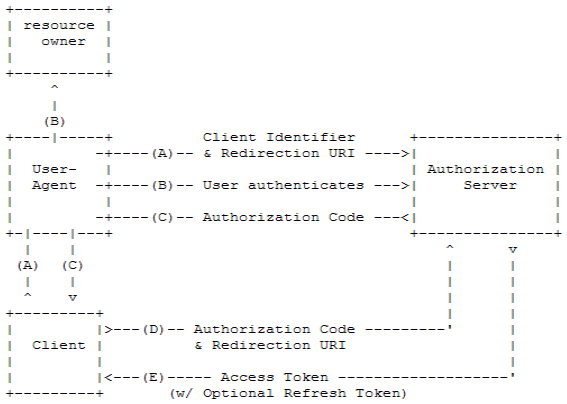
Section

Usage scenarios

Next, typical use cases of this new feature is discussed.

OAuth 2.0 usage scenario (1 of 4)

Authorization code flow



This slide represents an authorization code flow. In this scenario, the client initiates the flow by directing the resource owner's user-agent to the authorization endpoint. The client includes its client identifier, requested scope, local state, and a redirection URI to which the authorization server will send the user-agent back to once access is granted (or denied).

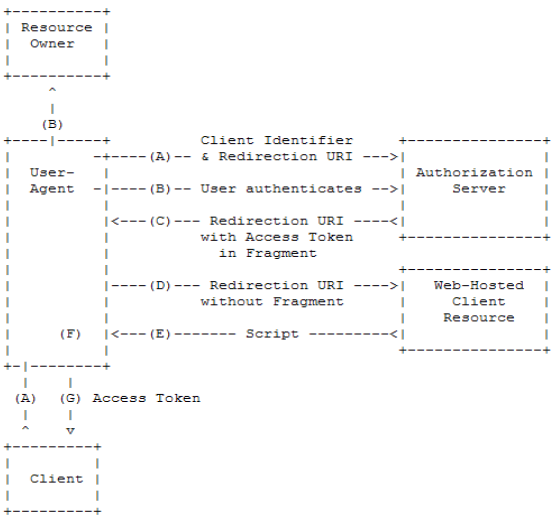
Next, the authorization server authenticates the resource owner (via the user-agent) and establishes whether the resource owner grants or denies the client's access request. Assuming the resource owner grants access, the authorization server redirects the user-agent back to the client using the redirection URI provided earlier (in the request or during client registration). The redirection URI includes an authorization code and any local state provided by the client earlier.

Then the client requests an access token from the authorization server's token endpoint by including the authorization code received in the previous step. When making the request, the client authenticates with the authorization server. The client includes the redirection URI used to obtain the authorization code for verification.

Finally, the authorization server authenticates the client, validates the authorization code, and ensures the redirection URI received matches the URI used to redirect the client in the step above. If valid, the authorization server responds back with an access token and optional refresh token.

OAuth 2.0 usage scenario (2 of 4)

Implicit grant



This slide represents an implicit grant. In this scenario the client initiates the flow by directing the resource owner's user-agent to the authorization endpoint. The client includes its client identifier, requested scope, local state, and a redirection URI to which the authorization server will send the user-agent back once access is granted (or denied).

Next, the authorization server authenticates the resource owner (via the user-agent) and establishes whether the resource owner grants or denies the client's access request. Assuming the resource owner grants access, the authorization server redirects the user-agent back to the client using the redirection URI provided earlier. The redirection URI includes the access token in the URI fragment.

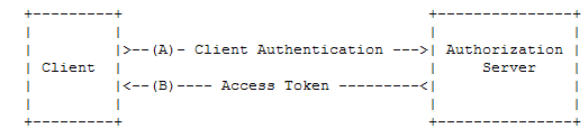
Then the user-agent follows the redirection instructions by making a request to the web-hosted client resource (which does not include the fragment). The user-agent retains the fragment information locally.

The web-hosted client resource returns a web page (typically an HTML document with an embedded script) capable of accessing the full redirection URI including the fragment retained by the user-agent, and extracting the access token (and other parameters) contained in the fragment.

Finally, the user-agent executes the script provided by the web-hosted client resource locally, which extracts the access token and passes it to the client.

OAuth 2.0 usage scenario (3 of 4)

Client credential



Client Credentials Flow

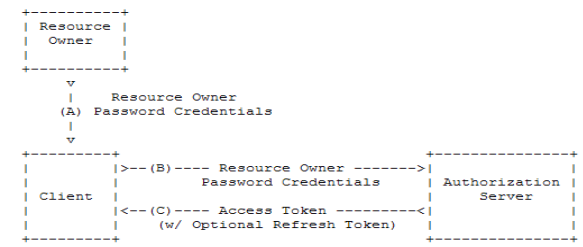
The flow illustrated includes the following steps:

(A) The client authenticates with the authorization server and requests an access token from the token endpoint.
(B) The authorization server authenticates the client, and if valid issues an access token.

The Client Credentials flow is used when the client is the resource owner. This flow is more like peer-to-peer than OAuth delegated authorization. The client accesses the token endpoint with the client ID and secret to be exchanged for an access token for future resource requests.

OAuth 2.0 usage scenario (4 of 4)

Resource owner password



Resource Owner Password Credentials Flow

The flow illustrated in Figure includes the following steps:

(A) The resource owner provides the client with its username and password.
(B) The client requests an access token from the authorization server's token endpoint by including the credentials received from the resource owner. When making the request, the client authenticates with the authorization server.

The Resource Owner Password Credentials flow passes the resource owner's user and password to the token endpoint. WebSphere application server provide a sample mediator to validate the user and password. This flow is mainly for integration of legacy environments.

Summary of features inside WebSphere

- WebSphere Application Server acts as OAuth Service Provider (SP) to handle OAuth 2.0 protocol requests.
- WebSphere Application Server acts as protected resource enforcement endpoint to authorize or deny request for deployed web resource.
- Allow multiple service providers co-existence.
- Allow administrator to revoke access tokens
- Allow client to revoke its authorization given by user.
- Optionally provide Subject for resource application to make authenticated downstream call or perform programmatic J2EE security.
- Support four typical OAuth 2.0 flows as defined in protocol.
- Support persistent OAuth services.

WebSphere Application server plays two roles, OAuth authorization service, and protected resource service.

WebSphere application server supports multiple OAuth authorization services and all four OAuth 2.0 flows: Authorization code flow, implicit flow, client credential flow, and resource owner password flow.

WebSphere also supports in memory and persistent services. With persistent OAuth service, OAuth token is saved to a data base table.

Beyond basic

- Multiple OAuth service providers coexist.
- WebSphere OAuth TAI supports multiple OAuth service providers.
- Persistent OAuth service
- Persistent OAuth clients in XML or in database.
- Support OAuth client revocation.
- Support all existing WebSphere web authentication mechanism for user authentication.

You can create many OAuth authorization services.

The WebSphere SAML TAI supports multiple OAuth authorization services. Each OAuth service can have its own processing rules.

OAuth access token can be saved to data base, so that a client's access token can survive server restart.

OAuth client can be saved to a XML in a simple format, or to a data base.

User authentication is not impacted by this feature. User can be authenticated with any existing authentication, like form login, SAML, or spnego.

References

- The OAuth 2.0 Authorization Framework
  - http://tools.ietf.org/html/draft-ietf-OAuth-v2-31
- The OAuth 2.0 Authorization Framework: Bearer Token Usage:
  - http://tools.ietf.org/html/draft-ietf-OAuth-v2-bearer-22

See these references for additional information about OAuth 2.0 and the supported bearer token.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WASV7.0.0.25_OAuth20.ppt

This module is also available in PDF format at: ../WASV7.0.0.25_OAuth20.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, disclaimer, and copyright information

IBM, the IBM logo, ibm.com, and WebSphere are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.