

Course Guide

AAA, OAuth, and OIDC in IBM DataPower V7.5

Course code WE753 / ZE753 ERC 1.0



April 2017 edition

Notices

This information was developed for products and services offered in the US.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

Trademarks

IBM, the IBM logo, and ibm.com 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 IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

© Copyright International Business Machines Corporation 2017.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Trademarks	vii
Course description	viii
Agenda	x
Unit 1. Authentication, authorization, and auditing (AAA)	1-1
How to check online for course material updates	1-2
Unit objectives	1-3
Authentication, authorization, and auditing	1-4
Authentication and authorization framework	1-5
AAA action and access control policy	1-7
How to define an access control policy (1 of 2)	1-8
How to define an access control policy (2 of 2)	1-9
Access control policy processing	1-10
Scenario 1: Authorize authenticated clients	1-11
Scenario 1: Sample SOAP request message	1-12
Scenario 1: Identify and authenticate the client	1-13
Scenario 1: Authorize access to resources	1-15
Scenario 2: Security token conversion	1-17
Scenario 2: Sample HTTP request message	1-18
Scenario 2: Identify and authenticate the client	1-19
Scenario 2: Authorize access to resources	1-20
Scenario 3: Multiple identity extraction methods	1-22
Scenario 3: Identify and authenticate the client	1-23
Scenario 3: LDAP details	1-24
Scenario 3: Authorize access to resources	1-25
Internal access control resources	1-26
AAA XML file	1-27
Example AAA XML file	1-28
Lightweight Third Party Authentication	1-29
External access control resource	1-30
Lightweight Directory Access Protocol	1-31
Security Assertion Markup Language	1-32
Types of SAML assertions	1-33
Scenario 4: Authorize valid SAML assertions	1-34
Scenario 4: SAML authentication statement (1 of 2)	1-35
Scenario 4: SAML authentication statement (2 of 2)	1-36
Scenario 4: SAML attribute statement (1 of 2)	1-37
Scenario 4: SAML attribute statement (2 of 2)	1-38
Scenario 4: Identify and authenticate the client	1-39
Scenario 4: Authorize access to resources	1-40
Scenario 4: Match SAML attributes	1-41
Access control policy by SAML information	1-42
Unit summary	1-43
Review questions	1-44
Review answers	1-45
Exercise: Configuring authentication and authorization in a service	1-46
Exercise objectives	1-47

Unit 2. OAuth overview and DataPower implementation	2-1
Unit objectives	2-2
What is OAuth?	2-3
Delegated authorization example	2-4
Example: Allow third-party access to social account	2-5
Example: Third-party access to online photo album	2-6
Before OAuth: Sharing user passwords	2-7
OAuth Step 1: Resource owner requests access	2-8
OAuth Step 2: OAuth client redirection to owner	2-9
OAuth Step 3: Authenticate owner with authorization server	2-10
OAuth Step 4: Ask resource owner to grant access to resources	2-11
OAuth Step 5: Resource owner grants client access to resources	2-12
OAuth Step 6: Authorization server sends authorization grant code to client	2-13
OAuth Step 7: Client requests access token from authorization server	2-14
OAuth Step 8: Authorization server sends authorization token to client	2-15
OAuth Step 9: OAuth client sends access token to resource server	2-16
OAuth Step 10: Resource server grants access to OAuth client	2-17
OAuth 2.0 protocol defines roles	2-18
OAuth 2.0 roles: Common implementation	2-19
OAuth 2.0 roles in the DataPower world (1 of 2)	2-20
OAuth 2.0 roles in the DataPower world (2 of 2)	2-21
Sample three-legged scenario in DataPower (1 of 4)	2-22
Sample three-legged scenario in DataPower (2 of 4)	2-23
Sample three-legged scenario in DataPower (3 of 4)	2-24
Sample three-legged scenario in DataPower (4 of 4)	2-25
Authorization request	2-26
Authorization response	2-27
Access token request	2-28
Access token response	2-29
Resource request	2-30
OAuth client and the OAuth Client Profile object	2-31
DataPower OAuth objects: OAuth Client (1 of 4)	2-32
DataPower OAuth objects: OAuth Client (2 of 4)	2-34
DataPower OAuth objects: OAuth Client (3 of 4)	2-35
DataPower OAuth objects: OAuth Client (4 of 4)	2-36
DataPower OAuth objects: OAuth Client Group	2-37
DataPower OAuth objects: Web Token Service	2-38
AAA policy: Key to OAuth behavior	2-39
AAA policy for the web token service	2-40
AAA policy for the resource server (1 of 2)	2-41
AAA policy for the resource server (2 of 2)	2-42
Unit summary	2-43
Review questions	2-44
Review answers	2-45
Exercise: Defining a three-legged OAuth scenario that uses DataPower services	2-46
Exercise objectives	2-47
Exercise overview: User interaction	2-48
Exercise overview: OAuth interaction (1 of 4)	2-49
Exercise overview: OAuth interaction (2 of 4)	2-50
Exercise overview: OAuth interaction (3 of 4)	2-51
Exercise overview: OAuth interaction (4 of 4)	2-52
Unit 3. Social Login support in DataPower	3-1
Unit objectives	3-2
Social Login and DataPower	3-3
What is OpenID Connect (OIDC)?	3-4

OIDC terms	3-5
ID token	3-6
ID token: Required claims	3-7
ID token: Example	3-8
Getting more information on the end-user	3-9
Some of the standard claims	3-10
Example flow for an end-user that must be authenticated	3-11
OIDC Step 1: End-user initiates request	3-12
OIDC Step 2: RP sends authorization request to the OP	3-13
OIDC Step 3: Authenticate end-user with the OP	3-14
OIDC Step 4: Ask end-user to give permission	3-15
OIDC Step 5: OP returns authorization response (authorization code) to the RP	3-16
OIDC Step 6: RP requests ID token, access token from OP	3-17
OIDC Step 7: OP returns access token/ID token	3-18
OIDC Optional Step: RP requests claims from the OP	3-19
OIDC support in DataPower V7.5.1	3-20
Options in the AAA Policy object: Identity extraction	3-21
Options in the AAA Policy object: Authenticate user	3-22
Options in the AAA Policy object: Postprocessing	3-23
Social login policy object	3-24
Social login policy configuration: Top	3-25
Social login policy configuration: Bottom	3-27
JWT validator object: Top of Main tab	3-28
JWT validator object: Bottom of Main tab	3-29
JWT validator object: Advanced tab	3-30
JWT generator object: Top of Main tab	3-31
JWT generator object: Bottom of Main tab	3-32
JWT generator object: Advanced tab	3-33
Google as the social login provider	3-34
Google as the social login provider: Social Login Policy	3-35
Where is the social login information in Google?	3-36
Getting Google's public certificate	3-37
Caching the Google's public certificate	3-38
Recovering the initial URI: The need	3-39
Recovering the initial URI: Example	3-40
Combining social login support (OIDC) with OAuth	3-41
Roles in the OAuth-OIDC combination	3-42
The "outer" OAuth flow	3-43
The "inner" OIDC flow	3-44
Combined flow (part 1 of 7)	3-45
Combined flow (part 2 of 7)	3-46
Combined flow (part 3 of 7)	3-47
Combined flow (part 4 of 7)	3-48
Combined flow (part 5 of 7)	3-49
Combined flow (part 6 of 7)	3-50
Combined flow (part 7 of 7)	3-51
Unit summary	3-52
Review questions	3-53
Review answers (1 of 2)	3-54
Review answers (2 of 2)	3-55
Exercise: Implementing an OIDC client	3-56
Exercise objectives	3-57
Unit 4. Course summary	4-1
Unit objectives	4-2
Course objectives	4-3

Lab exercise solutions	4-4
To learn more on the subject	4-5
Enhance your learning with IBM resources	4-6
Unit summary	4-7
Course completion	4-8
Appendix A. List of abbreviations	A-1

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

DataPower®

DB™

Domino®

Lotus®

Rational®

Tivoli®

WebSphere®

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware and the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the "Marks") of VMware, Inc. in the United States and/or other jurisdictions.

Social® is a trademark or registered trademark of TWC Product and Technology, LLC, an IBM Company.

Other product and service names might be trademarks of IBM or other companies.

Course description

AAA, OAuth, and OIDC in IBM DataPower V7.5

Duration: 1 day

Purpose

This course teaches you the developer skills that are required to configure and implement authentication and authorization support within your IBM DataPower Gateway V7.5 services.

A common requirement for DataPower services is to authenticate the sender of a message, and authorize that sender to request the message's behavior. The AAA action within DataPower provides the basics of the "authenticate, authorize, and audit" support.

OAuth is an authorization framework that defines a way for a client application to access server resources on behalf of another party. It provides a way for the user to authorize a third party to their server resources without sharing their credentials. DataPower supports OAuth specifications and protocols, and can provide an OAuth web token service.

OpenID Connect (OIDC) is an authentication layer that runs on top of an OAuth 2.0 authorization framework. DataPower can operate as an OIDC client.

In this course, you learn how to use the configuration options and processing actions to add the AAA support to a service, implement an OAuth 2.0 scenario, and add OIDC support.

Hands-on exercises give you experience working directly with an IBM DataPower gateway. The exercises focus on skills such as configuring a AAA action, configuring a web token service, and creating an OIDC client.

Audience

This course is designed for integration developers who configure service policies on IBM DataPower Gateways.

Prerequisites

Before taking this course, you should successfully complete course *Essentials of Service Development for IBM DataPower Gateway V7.5* (WE751G) or *Essentials of Service Development for IBM DataPower Gateway V7.5* (ZE751G). You should also be familiar with AAA, OAuth 2.0, and OIDC concepts.

Objectives

- Describe the AAA framework within the IBM DataPower Gateway
- Explain the purpose of each step in an access control policy

- Configure a AAA action to enforce authentication and authorization policies that are in a AAA information file and in an LDAP server
- Describe the OAuth 2.0 framework
- Explain the role that a DataPower gateway performs in an OAuth 2.0 framework
- Configure the DataPower objects that are used for OAuth 2.0 interactions
- Define Social Login
- Describe how to configure Social Login in DataPower
- Configure an OIDC client

Agenda

**Note**

The following unit and exercise durations are estimates, and might not reflect every class experience.

Day 1

- (00:15) Course introduction
- (01:00) Unit 1. Authentication, authorization, and auditing (AAA)
- (00:45) Exercise 1. Configuring authentication and authorization in a service
- (00:45) Unit 2. OAuth overview and DataPower implementation
- (01:30) Exercise 2. Defining a three-legged OAuth scenario that uses DataPower services
- (00:45) Unit 3. Social Login support in DataPower
- (01:00) Exercise 3. Implementing an OIDC client
- (00:15) Unit 4. Course summary

Unit 1. Authentication, authorization, and auditing (AAA)

Estimated time

01:00

Overview

This unit describes the authentication, authorization, and auditing (AAA) framework within the IBM DataPower Gateway. These three facets of security both monitor and restrict access to resources.

How you will check your progress

- Checkpoint
- Hands-on exercise

References

IBM DataPower Gateway Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.5.0

How to check online for course material updates



Note: If your classroom does not have internet access, ask your instructor for more information.

Instructions

1. Enter this URL in your browser:
ibm.biz/CloudEduCourses
2. Find the product category for your course, and click the link to view all products and courses.
3. Find your course in the course list and then click the link.
4. The wiki page displays information for the course. If the course has a corrections document, this page is where it is found.
5. If you want to download an attachment, such as a course corrections document, click the **Attachments** tab at the bottom of the page.

Comments (0)	Versions (1)	Attachments (1)	About
--------------	--------------	------------------------	-------

6. To save the file to your computer, click the document link and follow the prompts.

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-1. How to check online for course material updates

Unit objectives

- Describe the AAA framework within the DataPower Gateway
- Explain the purpose of each step in an access control policy
- Authenticate and authorize requests with:
 - WS-Security Username and binary security tokens
 - HTTP Authorization header claims
 - Security Assertion Markup Language (SAML) assertions

Authentication, authorization, and auditing

- In the DataPower gateway, AAA represents three security processes: **authentication, authorization, and auditing**



– **Authentication**
verifies the identity
of the request
sender



– **Authorization**
determines whether
the client has
access to the
requested resource



– **Auditing** keeps
records of any
attempts to
access resources

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

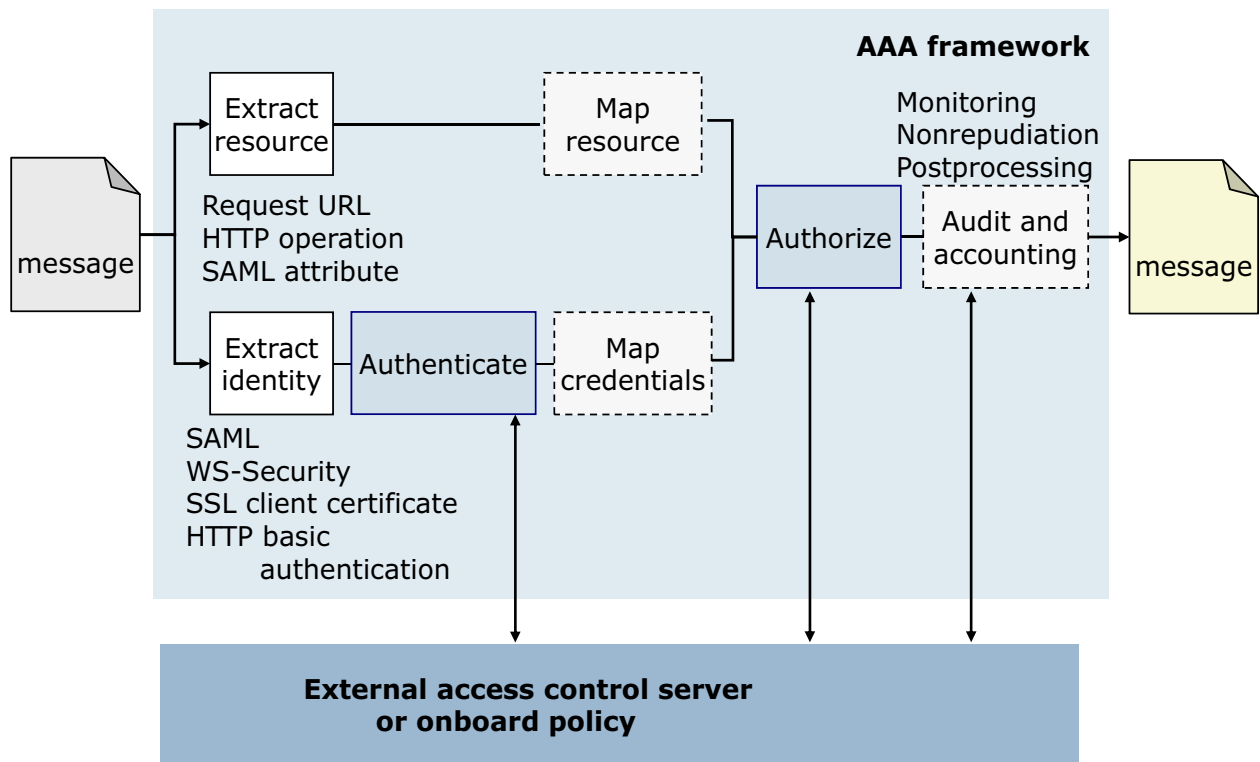
Figure 1-3. Authentication, authorization, and auditing

Authentication always precedes authorization. A policy cannot decide whether a request proceeds if it does not know the identity of the requester. For example, a security guard first determines whether someone is an employee of the company. After this step, the guard determines whether that employee has access to the building. Together, authentication and authorization restrict access to resources.

Although auditing does not directly protect resources against unauthorized access, this third process has an important role in securing resources. A record of successful and unsuccessful access attempts allows the security infrastructure to detect suspicious activity in the system. Historical logs also enforce nonrepudiation; clients cannot deny accessing the system in the past.

In literature these three steps are commonly known as “AAA,” which is pronounced “triple A.”

Authentication and authorization framework



Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-4. Authentication and authorization framework

The AAA action combines three security processes into a single stylesheet transform. In the first step, the stylesheet extracts the identity token from the message. To verify the claims that the token makes, the stylesheet either authenticates the token against an on-board policy or queries an external access control server. As soon as the client identity is confirmed, the stylesheet maps the client credentials to one of the users or groups that the service defines.

In the second step, the stylesheet extracts the requested resource from the message. For web services, a resource represents a service or service operation. If the requested resource is an alias for one or more back-end resources, the stylesheet maps the alias to the actual resource names as well.

When the stylesheet determines the requested back-end resource and confirms the client identity, it decides whether the client has permission to access the requested resource. In other words, the stylesheet authorizes access to a back-end resource.

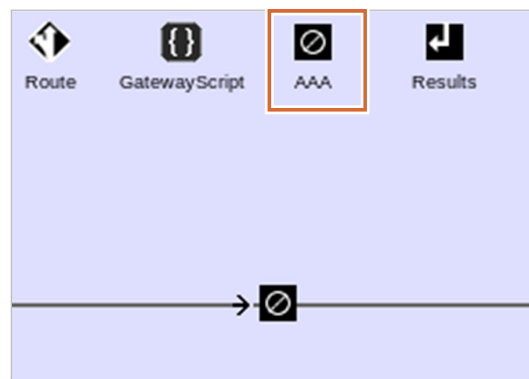
The final step is auditing and accounting. The stylesheet records any access attempts, successful or unsuccessful, for monitoring and nonrepudiation. The stylesheet can also do postprocessing steps, such as generating various tokens for the outgoing message. A custom stylesheet can also be specified.

The various tokens that can be generated are:

- SAML assertion
- WS-Security Kerberos AP-REQ
- WS-Security user name
- LTPA
- Kerberos SPNEGO
- ICRX
- JSON web token (JWT)

AAA action and access control policy

- To restrict access to resources, add a **AAA action** to a document processing rule
 - AAA action invokes an **access control policy**, or **AAA policy**
- An **access control policy**, or a **AAA policy**, determines whether a requesting client is granted access to a specific resource
 - These policies are filters that accept or deny specific client requests



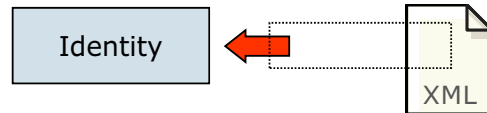
Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-5. AAA action and access control policy

How to define an access control policy (1 of 2)

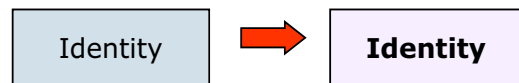
1. Define one or more identity extraction methods



2. Define the authentication method



3. Map authentication credentials (optional)



Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-6. How to define an access control policy (1 of 2)

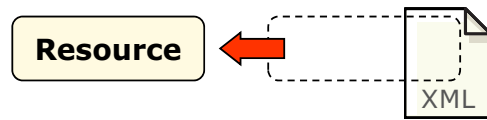
The access control policy steps relate directly to the processing stages within the AAA framework. In the first step, the policy defines how the framework retrieves information about the client identity. The framework can treat the requested URL, the client IP address, the HTTP header, or any part of the message, as a client identifier. When it is extracted, the second step describes how to verify the claimed identity that is stored in the message. If the authorization method (which is described on the next slide) expects a different client identifier, the policy can apply a custom stylesheet to convert the authentication credentials.

The authentication credential mapping step is optional.

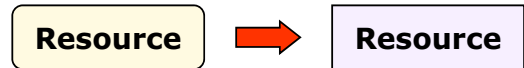
The colors and shapes of the identity token uniquely represent different security token types.

How to define an access control policy (2 of 2)

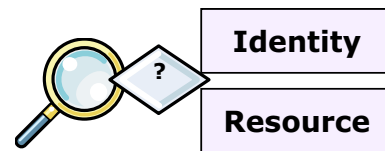
4. Define resource extraction methods



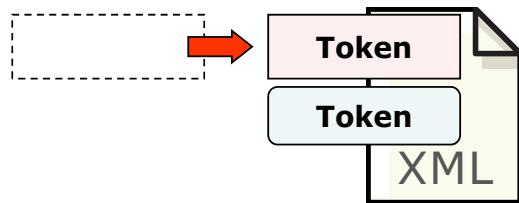
5. Map requested resources (optional)



6. Define the authorization method



7. Specify postprocessing actions (optional)



Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

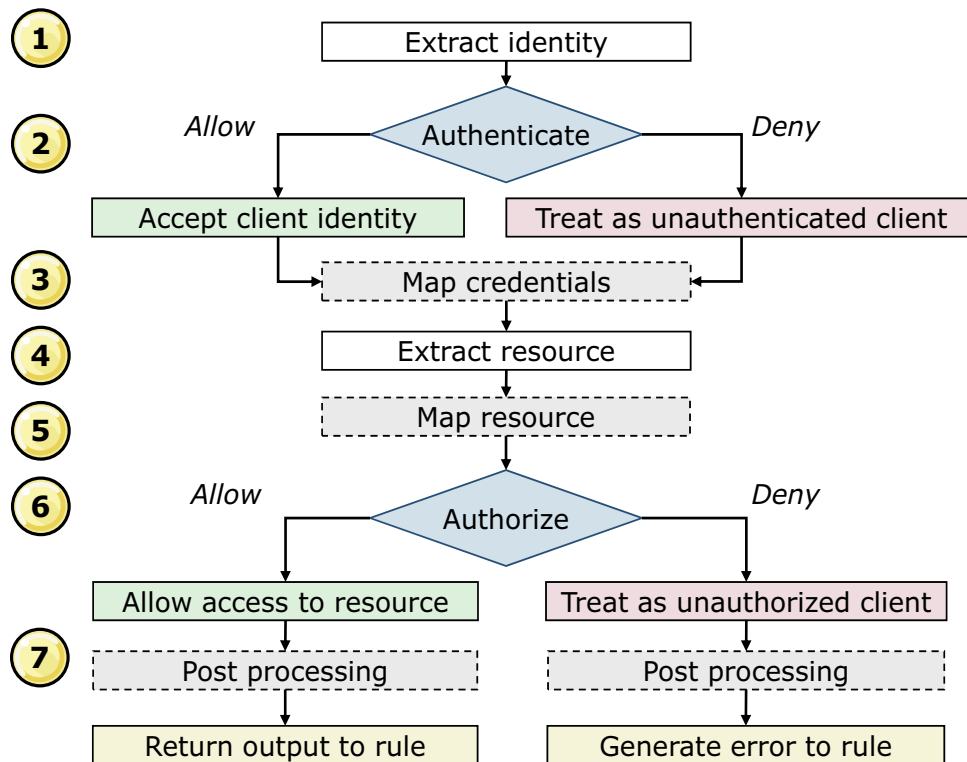
Figure 1-7. How to define an access control policy (2 of 2)

If the authentication step succeeds, the policy determines the resources that the client requests before making a final decision on whether to authorize access. An optional mapping step matches the resource request with a type expected by the authorization method.

When authentication and authorization succeed, monitoring and postprocessing steps can take place. The monitoring step records whether the access control policy as a whole succeeds or fails. Such information can be used for auditing purposes. Unlike the monitoring step, post processing occurs only if the policy authorizes the resource request.

The postprocessing step can add more security tokens such as a SAML assertion, WS-Security Username token, LTPA token, Kerberos SPNEGO token, and a JSON web token (JWT).

Access control policy processing



Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-8. Access control policy processing

The numbers correspond to the access control policy steps detailed on the previous two slides. Keep in mind that the output message is returned to the processing rule, not back to the actual client itself. Similarly, an **On Error action** or an **error rule** suppresses or handles errors that are generated from a AAA action.

The only part of the postprocessing step that occurs when authorization fails is the incrementing of the authorization failures counter (if one exists).

Within the postprocessing step, monitors track the requests.

Scenario 1: Authorize authenticated clients

- Create an access control policy that handles client SOAP web service requests with the following conditions:
 - The client communicates to the DataPower gateway over a Secure Sockets Layer (SSL) connection
 - A WS-Security UsernameToken element holds the requesting client identity
 - Verifies the claimed identity of the client against a list that is stored on the DataPower gateway itself
 - The requested resource is the web service operation
 - Allows any authenticated client access to the web service operation

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-9. Scenario 1: Authorize authenticated clients

In this scenario, the client includes a WS-Security user name token with a password or password digest as a proof of identity. As a good practice, clients send plain text tokens, such as the WS-Security user name token, within a secure channel, such as an SSL connection.

The access control policy on the DataPower gateway verifies the user name and password against a built-in user list. It assumes that all authenticated users have full access to any resource protected by the policy.

Scenario 1: Sample SOAP request message

```
<?xml version="1.0" encoding="UTF-8">
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:wsse="http://...wssecurity-secext-1.0.xsd"
  xmlns:q0="http://east.address.training.ibm.com">
  <soap:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>Alice</wsse:Username>
        <wsse:Password>ond3mand</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <q0:retrieveAll />
  </soap:Body>
</soap:Envelope>
```

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-10. Scenario 1: Sample SOAP request message

The WS-Security user name token provides a basic method to transport user credentials to a web service. The password field can be in plain text, or the Secure Hash Algorithm (SHA1) can hash it. Because SHA1 is a well-known algorithm, a hashed password provides a minimal level of security by obfuscating the password. Messages with these identity credentials are sent only over a secure connection.

For the sake of brevity, the URI for the **wsse** namespace declaration is truncated. For the URI, see the WS-Security V1.1 specification.

Within the SOAP body, the child element describes the requested web service operation. In effect, this element identifies the resource that is requested in this call.

Two elements are highlighted in dark blue: the WS-Security header and the `retrieveAll` web service operation call.

Scenario 1: Identify and authenticate the client

1. Create a AAA policy object on the DataPower gateway
2. Extract the client's identity with the **Password-carrying UsernameToken Element from WS-Security header** option
3. For the authentication method,
Use AAA information file
 - Specify the name of the AAA information file in the **URL** field
4. Leave the identity mapping method at **None**

Define how to map credentials.

Method	None
---------------	------

Define how to extract a user's identity from an incoming request.

<input type="checkbox"/>	HTTP Authentication header
<input checked="" type="checkbox"/>	Password-carrying UsernameToken element from WS-Security header
<input type="checkbox"/>	Derived-key UsernameToken element from WS-Security header

Define how to authenticate the user.

<input type="radio"/>	Accept LTPA token
<input type="radio"/>	Accept SAML assertion with valid signature
<input type="radio"/>	Bind to LDAP server
<input type="radio"/>	Contact CA Single Sign-On (formerly Netegrity SiteMinder)
<input type="radio"/>	Contact ClearTrust server
<input type="radio"/>	Contact IBM Security Access Manager
<input type="radio"/>	Contact NSS for SAF authentication
<input type="radio"/>	Contact SAML server for SAML Authentication statement
<input type="radio"/>	Contact WS-Trust server for WS-Trust token
<input type="radio"/>	Custom template
<input type="radio"/>	Pass identity token to authorization phase
<input type="radio"/>	Retrieve SAML assertions that corresponds to SAML Browser Artifact
<input checked="" type="radio"/>	Use AAA information file
<input type="radio"/>	Use certificate from BinarySecurityToken

URL

store:///	
AAAInfo.xml	+

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-11. Scenario 1: Identify and authenticate the client

The choices for identity extraction are:

- HTTP Authentication header
- Password-carrying UsernameToken element from WS-Security header
- Derived-key UsernameToken element from WS-Security header
- BinarySecurityToken element from WS-Security header
- WS-SecureConversation identifier
- WS-Trust Base or Supporting token
- Kerberos AP-REQ from WS-Security header
- Kerberos AP-REQ from SPNEGO token
- Subject DN of SSL certificate from connection peer
- Name from SAML Attribute assertion
- Name from SAML Authentication assertion
- SAML Artifact
- Client IP address

- Subject DN from certificate in message signature
- Token extracted from message
- Token extracted as cookie value
- LTPA token
- Processing metadata
- JWT
- Custom processing
- HTML forms-based authentication
- Redirect to a social login provider
- OAuth

The choices for authentication method are:

- Accept LTPA token
- Accept SAML assertion with valid signature
- Bind to LDAP server
- Contact CA Single Sign-On (formerly Netegrity SiteMinder)
- Contact ClearTrust server
- Contact IBM Security Access Manager
- Contact NSS for SAF authentication
- Contact SAML server for SAML Authentication statement
- Contact WS-Trust server for WS-Trust token
- Custom template
- Pass identity token to authorization phase
- Retrieve SAML assertions that corresponds to SAML Browser Artifact
- Use AAA information file
- Use certificate from BinarySecurityToken
- Use established WS-SecureConversation security context
- Use RADIUS server
- Use verified JWT, access token, or ID token
- Validate Kerberos AP-REQ for server principal
- Validate signer certificate for digitally signed message
- Validate SSL certificate from connection peer

Scenario 1: Authorize access to resources

5. Select **Local name of request element** as the resource extraction method

- The name of the child element in the SOAP body of the request is the request element name

6. Leave the resource mapping method at **None**

7. For the authorization method, allow any request from an authenticated client to proceed

The screenshot shows the configuration interface for authorizing access to resources. It is divided into three main sections:

- Resource Identification Methods:** A list of methods with checkboxes. The 'Local name of request element' option is selected (checked).
- Define how to map resources:** A section with a 'Method' dropdown menu set to 'None'.
- Define how to authorize a request:** A list of authorization methods with radio buttons. The 'Allow any authenticated client' option is selected.

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-12. Scenario 1: Authorize access to resources

The authorization choices are:

- AAA information file
- Allow any authenticated client
- Always allow
- Check membership in LDAP group
- Contact CA Single Sign-On (formerly Netegrity SiteMinder)
- Contact ClearTrust server
- Contact IBM Security Access Manager
- Contact NSS for SAF authorization
- Contact OAuth STS
- Custom template
- Generate SAML Attribute query
- Generate SAML Authorization query
- Use SAML attributes from authentication
- Use XACML Authorization decision

- Use XACML Authorization decision

Scenario 2: Security token conversion

- Create an access control policy that handles client SOAP web service requests with the following conditions:
 - The client communicates to the DataPower gateway over a Secure Sockets Layer (SSL) connection
 - The HTTP BASIC-AUTH header information holds the identity of the requesting client
 - Generates a WS-Security UsernameToken element corresponding to the HTTP BASIC-AUTH header
 - Defers the authentication and authorization tasks to the back-end web service

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-13. Scenario 2: Security token conversion

HTTP BASIC-AUTH is the basic authentication scheme. See the following slide for an example of an HTTP request message with a basic authentication header.

Scenario 2: Sample HTTP request message

```
POST /EastAddress/services/AddressSearch HTTP/1.1
Host: www.example.com
Content-type: text/xml; charset=utf-8
Content-length: 237
Authorization: Basic T3phaXI6U2hlaWtoTkJha2U=

<?xml version="1.0" encoding="UTF-8">
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:q0="http://east.address.training.ibm.com">
  <soap:Header />
  <soap:Body>
    <q0:retrieveAll />
  </soap:Body>
</soap:Envelope>
```

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-14. Scenario 2: Sample HTTP request message

In this scenario, the HTTP authorization header field is used for authentication. Remember that the client encoded the user name and password in Base64. This encoding method is known, hence, the client uses an SSL connection to keep the contents of this message private.

Base64 is a binary-to-text encoding scheme by printable (mostly alphanumeric) characters. As an MIME content transfer encoding, it is used to encode binary data into email messages.

In the HTTP basic authentication scheme, the user name and password are concatenated with a colon (:) before it is encoded by Base64. For example, the user name “Alice” and the password “ond3mand” become “Alice:ond3mand”. In Base64 encoding, the user name and password string is “QWxpY2U6b25kM21hbmQ=”

Scenario 2: Identify and authenticate the client

1. Create a AAA policy object on the DataPower gateway
2. Extract the client's identity with the **HTTP Authentication header** option
 - The value within the Authorization HTTP header represents the HTTP authentication header
3. For the authentication method, specify **Pass identity token to authorization phase**
4. Leave the identity mapping method at **none**

Define how to extract a user's identity from an incoming request.

☒ HTTP Authentication header

☐ Password-carrying UsernameToken element from WS-Security header

☐ Derived key UsernameToken element from WS-Security header

☐ Contact SAML server for SAML Authentication statement

☐ Contact WS-Trust server for WS-Trust token

☐ Custom template

☒ Pass identity token to authorization phase

☐ Retrieve SAML assertions that corresponds to SAML Browser Artifact

☐ Use AAA information file

☐ Use certificate from BinarySecurityToken

Define how to map credentials.

Method | *

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-15. Scenario 2: Identify and authenticate the client

Scenario 2: Authorize access to resources

5. Select **Local name of request element** as the resource extraction method
 - The name of the child element in the SOAP body of the request is the request element name
6. Leave the resource mapping method at **None**
7. Set the authorization method to always allow requests
8. In the postprocessing step, add the WS-Security Username Token

The screenshot shows the configuration interface for authorizing access to resources. It is divided into two main sections:

- Resource Identification Methods:** This section contains four radio button options:
 - ☐ URL sent by client
 - ☐ URI of top level element in message
 - ☒ Local name of request element
 - ☐ HTTP operation (GET or POST)
- Define how to authorize a request:** This section contains five radio button options:
 - ☐ AAA information file
 - ☐ Allow any authenticated client
 - ☒ Always allow
 - ☐ Check membership in LDAP group
 - ☐ Contact ClearTrust server

Below these sections is a third section titled **Choose any post processing.** which includes several toggle switches and a dropdown menu:

- Run Custom Post Processing:** A toggle switch set to 'off'.
- Add WS-Security UsernameToken:** A toggle switch set to 'on'.
- Include Password:** A toggle switch set to 'on'.
- WS-Security UsernameToken Password Type:** A dropdown menu currently showing 'Digest'.

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-16. Scenario 2: Authorize access to resources

The **Run Custom Post Processing** setting applies a custom stylesheet or GatewayScript to the outgoing request message. This setting does not require enablement for a built-in postprocessing step, such as adding a WS-Security user name token.

The added WS-Security Username token for a username of “student”, a password of “web1sphere”, and a password type of “Text” is:

```
<wsse:UsernameToken>
<wsse:Username>student</wsse:Username>
<wsse:Password
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profi
le-1.0#PasswordText">web1sphere</wsse:Password>
<wsse:Nonce
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message
-security-1.0#Base64Binary">N2FmMzg3OTFtZDI4OS00MzgzLTNmYWUtNzY3MzZhYmRmM2Zh</wsse
:Nonce>
<wsu:Created
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-util
ity-1.0.xsd"
>2015-03-24T21:16:44Z</wsu:Created>
</wsse:UsernameToken>
```

The same token, with a password type of “Digest” is:

```
<wsse:UsernameToken>
<wsse:Username>student</wsse:Username>
<wsse:Password
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profi
le-1.0#PasswordDigest">JDqa4I7DaWicLrj+ykiSBQT0MFc=</wsse:Password>
<wsse:Nonce
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message
-security-1.0#Base64Binary">YzVhNmQ3OTctZmE5Mi00NjdhLWFiYmYtZDUyNDVmNmRjNTEw</wsse
:Nonce>
<wsu:Created
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-util
ity-1.0.xsd"
>2015-03-24T21:25:12Z</wsu:Created>
</wsse:UsernameToken>
```

In both cases, the optional <Nonce> and <Created> elements are specified. These elements and the password can be used to protect against replay attacks.

Scenario 3: Multiple identity extraction methods

- Create an access control policy that handles client SOAP web service requests with the following conditions:
 - Uses either a WS-Security UsernameToken element or a BinarySecurityToken element from the WS-Security header to determine the client's identity
 - Verifies the identity of the client
 - The requested resource is the web service operation
 - Allows any authenticated client access to the web service operation

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-17. Scenario 3: Multiple identity extraction methods

For identity extraction methods, the policy runs *all* checked methods. The system runs the methods in the order that is presented in the check box list. Afterward, the system concatenates all identities that are found for authentication. This scheme allows different clients to use different identification methods.

However, if a client includes more than one identifier in the message, *both* identifiers must pass the authentication stage.

Choosing more than one identity extraction method allows an access control policy to support different credentials from different clients. In this example, some clients support XML signatures, but not WS-Security. Other clients support the WS-Security standard. The former adds an XML signature block directly into the SOAP header, while the latter adds a binary security token within a WS-Security header.

Scenario 3: Identify and authenticate the client

1. Create a AAA policy object on the DataPower gateway
2. Extract the client's identity from the Username element or a BinarySecurityToken
 - Separate WS-Security token profiles describe the structure of the UsernameToken and the BinarySecurityToken
3. For the authentication method, specify **Bind to LDAP server**
 - The LDAP directory server provides an external list of authenticated users
4. Leave the identity mapping method at **none**

Define how to extract a user's identity from an incoming request.

- ☐ HTTP Authentication header
- ☒ Password-carrying UsernameToken element from WS-Security header
- ☐ Derived-key UsernameToken element from WS-Security header
- ☒ BinarySecurityToken element from WS-Security header
- ☐ WS-SecureConversation identifier
- ☐ WS-Trust Base or Supporting token

Define how to authenticate the user.

- ☐ Accept LTPA token
- ☐ Accept SAML assertion with valid signature
- ☒ Bind to LDAP server
- ☐ Contact ClearTrust server
- ☐ Contact IBM Security Access Manager

Define how to map credentials.

Method *

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-18. Scenario 3: Identify and authenticate the client

The **Bind to LDAP server** choice causes the page to redisplay with the LDAP options on the lower part of the page. The lower part is shown on the next slide.

Scenario 3: LDAP details

When connecting to LDAP, further details are needed

1. Specify the LDAP server URL and port
2. Indicate the **LDAP Bind DN** and **LDAP Bind Password Alias** for the LDAP query
3. Use the **LDAP Search Attribute** fields to verify the password digest from a WS-Security Username Token
4. Use the **LDAP Prefix** and **LDAP Suffix** fields to build the LDAP query
 - For example, the extracted identity of **John** would result in a distinguished name of **cn=John,dc=ibm,dc=com**

The screenshot shows the 'LDAP Load Balancer Group' configuration page. It includes fields for Host, Port (389), SSL Type (Client Profile), SSL Client Profile (none), LDAP Bind DN, LDAP Bind Password Alias (none), LDAP Search Attribute (userPassword), LDAP Version (v3), LDAP Search for DN (radio buttons for on/off, with 'off' selected), LDAP Prefix (cn=), LDAP Suffix (dc=ibm,dc=com), User auxiliary LDAP attributes, and LDAP Read Timeout (60). Numbered callouts are placed over the following fields: 1 over Host, 2 over LDAP Bind DN, 3 over LDAP Search Attribute, and 4 over LDAP Prefix.

Authentication, authorization, and auditing (AAA)

Figure 1-19. Scenario 3: LDAP details

© Copyright IBM Corporation 2017

If **Bind to LDAP server** is selected, the page repaints to display entry fields for the LDAP details.

The targeted LDAP server can be a load balancer group that is composed of multiple LDAP servers.

The DataPower to LDAP server connection is usually over an SSL connection. For example, when you specify an **SSL Type** of "Client Profile", the following field asks for the **SSL Client Profile** object.

The **LDAP Bind Password Alias** identifies an object that contains the text of the password. Hence, the configuration of the LDAP server connection does not directly expose the Bind password.

Scenario 3: Authorize access to resources

5. Select **Local name of request element** as the resource extraction method

- The name of the child element in the SOAP body of the request is the request element name

6. Leave the resource mapping method at **none**

7. For the authorization method, allow any request from an authenticated client to proceed

Resource Identification Methods	<input type="radio"/> URL sent by client
	<input type="radio"/> URI of top level element in message
	<input checked="" type="radio"/> Local name of request element
	<input type="radio"/> HTTP operation (GET or POST)

Define how to map resources.	
Method	none *

Define how to authorize a request.
<input type="radio"/> AAA information file <input checked="" type="radio"/> Allow any authenticated client <input type="radio"/> Always allow <input type="radio"/> Check membership in LDAP group

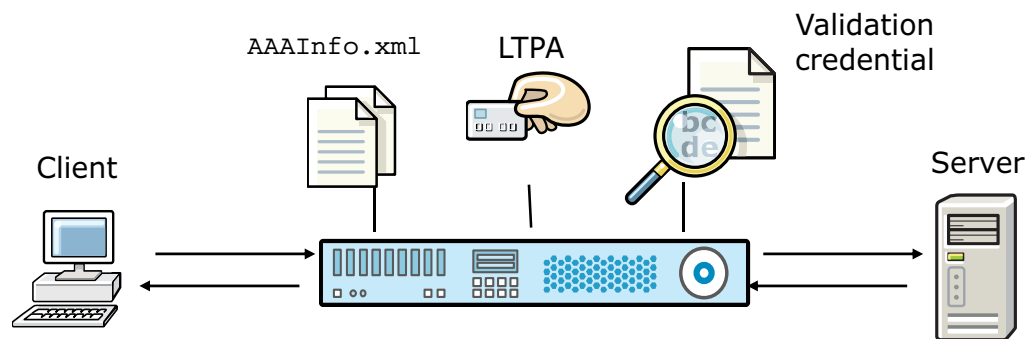
Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-20. Scenario 3: Authorize access to resources

Internal access control resources

- Authentication and authorization can be performed on the DataPower box by:
 - AAA file: XML file that contains validation information for the AAA steps (authenticate, authorize, map credentials, map resource)
 - LTPA: Token type that the IBM WebSphere Application Server and Lotus Domino products use
 - Validation credential object: List of certificates that are used to validate the incoming digital signature



Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

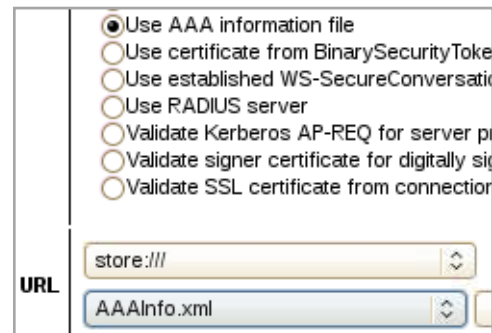
Figure 1-21. Internal access control resources

Lightweight Third-Party Authentication (LTPA) is an authentication technology used in IBM WebSphere and Lotus Domino products.

The validation credential object references a list of certificates on the gateway that validate the incoming digital signature. This object is also used when configuring client-side SSL.

AAA XML file

- The AAA XML file is used to validate the credentials in a AAA policy
- Used by the following AAA steps:
 - Authenticate
 - Authorize
 - Map credentials
 - Map resource
- Useful for testing of AAA policy when off-box resources not available
 - Use in production to maintain small list of AAA credentials
- For the authenticate or authorize step in the AAA policy, select **Use AAA information file**
 - Select an existing XML file or create a AAA file



Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-22. AAA XML file

A sample AAA information file is in store:///AAAInfo.xml.

Example AAA XML file

```
<aaa:AAAInfo xmlns:aaa="http://www.datapower.com/AAAInfo">
  <aaa:FormatVersion>1</aaa:FormatVersion>
  <aaa:Filename>local:///AddressInfo.xml</aaa:Filename>
  <aaa:Summary>
    AAA file to validate credentials for Address users
  </aaa:Summary>

  <aaa:Authenticate>
    <aaa:Username>AddressAdmin</aaa:Username>
    <aaa:Password>password</aaa:Password>
    <aaa:OutputCredential>
      AddressUser
    </aaa:OutputCredential>
  </aaa:Authenticate>
</aaa:AAAInfo>
```

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-23. Example AAA XML file

The Authenticate step uses this AAA XML file to validate the extracted identity. The incoming identity has a user name of **AddressAdmin** and password of **password**.

After the Authenticate step, **AddressUser** is returned as the output credential.

Lightweight Third Party Authentication

- Lightweight Third Party Authentication (LTPA) is a single sign-on (SSO) credential format for distributed, multiple application server environments
 - LTPA is a proprietary token type that the IBM WebSphere Application Server and Lotus Domino products use
- The purpose of LTPA is threefold:
 - Propagates the caller identity through a unique identifier of the client
 - Establishes a trust relationship between two servers, with one as the client and one as the server, through a signed token
 - Keeps the information within the token secret by signing and encrypting the token
 - A set of key files must be uploaded to the DataPower gateway to decrypt and validate the digital signature within the token

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-24. Lightweight Third Party Authentication

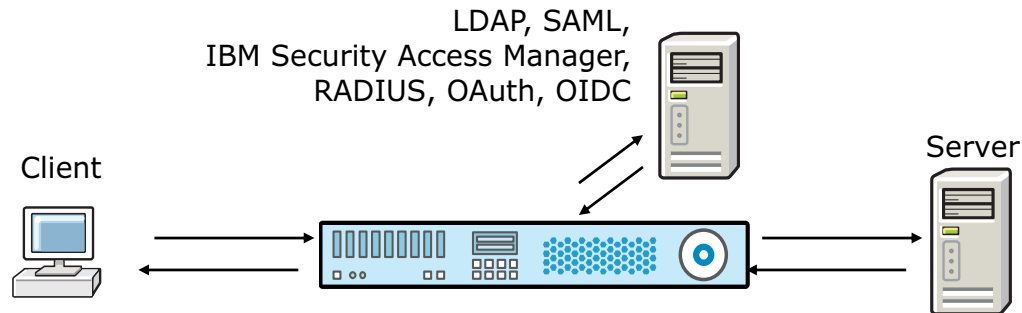
For more information, see the article *WS-Policy security integration between DataPower and WebSphere Application Server*, which includes a section on using the LTPA token:

http://www.ibm.com/developerworks/websphere/library/techarticles/0911_rasmussen/0911_rasmussen.html.

Also, see the **LTPA** section of the Knowledge Center:

https://www.ibm.com/support/knowledgecenter/SS9H2Y_7.5.0/com.ibm.dp.doc/ltpa_introduction.html.

External access control resource



- Delegates the authentication and authorization task to an external security system
- The authentication and authorization tasks can be delegated to the same system or to separate systems
 - For example, an LDAP directory tracks client identities, while IBM Security Access Manager determines whether the client has access to the specified resource
 - The **map credentials** and **map resource** steps convert the security token to match the input that the authorization step requires

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-25. External access control resource

It is also possible to do authentication and authorization on an IBM Security Access Manager system. IBM Security Access Manager can be configured to use its own user repository for authentication instead of using a separate, external Lightweight Directory Access Protocol (LDAP) server.

The list of external access controls on this slide is merely an example. For a full list of security products and specifications that are supported, see the product documentation.

OAuth and OpenID Connect (OIDC) are covered in later units.

Delegating access control to a separate server makes it easier to enforce a uniform security policy across the entire system.

Lightweight Directory Access Protocol

- LDAP provides a means of storing and retrieving information about people, groups, or objects on a centralized X.500 or LDAP directory server
 - X.500 enables the information to be organized and queried, by LDAP, from multiple web servers by various attributes
 - LDAP reduces system resources by including only a functional subset of the original X.500 Directory Access Protocol (DAP)
- A few facts about LDAP:
 - An LDAP directory is a tree of directory entries
 - The **distinguished name** (DN) is a unique identifier for entries
 - A **bind** operation authenticates the client by sending the client's distinguished name and password in cleartext
 - Use an SSL connection to keep LDAP queries secret

Security Assertion Markup Language

- SAML provides an XML-based framework for exchanging authentication, authorization, and attribute assertions between the entities
 - Provides a standard, platform-neutral way for exchanging security information between a security system and an application that trusts the security system
 - Expands the authentication and authorization trust model from existing systems by allowing new systems to delegate trust management to other systems
 - Includes protocol for requesting this information from security authorities
 - For example, SOAP and HTTP bindings

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-27. Security Assertion Markup Language

Federated security systems require an interoperable way of sending security information from one system to another. The Security Assertion Markup Language (SAML) is designed specifically for this purpose. It is analogous to how the SOAP specification defines a messaging model for transferring information between web service clients and servers.

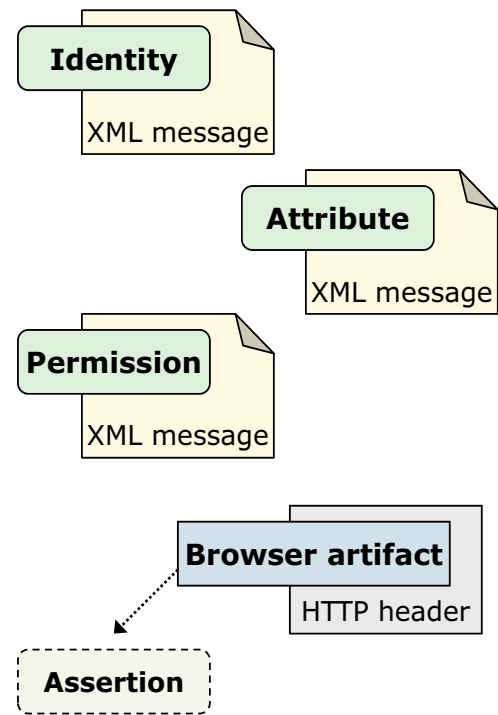
SAML allows clients or intermediaries to embed claims, or assertions, into the message. One common use for assertions is single sign-on: after a security server authenticates a client, a SAML authentication statement is tagged to the client request. Subsequent systems that process the request need only to trust the assertion instead of authenticating the client again.

SAML does not create another language for writing authentication or authorization tokens. Its assertions are self-describing wrappers around any token type. This concept is similar to how the web services security standard describes a uniform method for storing binary security tokens in a SOAP message. The actual binary security token is an implementation detail beyond these specifications.

SAML does not require storage within a WS-Security header. It is possible to have a SAML assertion directly on the SOAP header. SAML V1.1 defines how assertions look within a WS-Security header.

Types of SAML assertions

- Three main types of XML-based SAML assertions exist:
 - **Authentication** assertions represent the identity of the specified subject that another entity verifies
 - **Attribute** assertions represent any attributes that are associated with the specified subject
 - **Authorization** decision assertions represent whether the specified subject is granted or denied access to a specified resource
- In addition, the HTTP binding provides a non-XML reference:
 - A *SAML artifact that is embedded* in the URL query string provides a reference to an actual SAML assertion that is stored in a remote site



Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-28. Types of SAML assertions

In plain terms, here are some typical statements that the three types of SAML assertions make:

- Authentication statement: “I am Bob Smith.”
- Attribute statement: “Bob Smith is a payroll manager.”
- Authorization decision statement: “Payroll managers can run the Payroll Update web service.”

These assertions avoid repeating the same checks on the same message as it passes through different systems. In addition, assertion statements delegate the authentication and authorization task to a separate server.

The last point describes the HTTP binding for SAML. Remember that SAML is not only used for web services. For example, a web application server might want to verify a SAML assertion in a single sign-on (SSO) scenario. Without even examining the HTTP request message, the server extracts and dereferences a SAML assertion from the URL query string.

The common practice is to embed a SAML assertion inside a SOAP header. However, the IBM DataPower gateway can also dereference browser artifacts that point to a SAML assertion that is kept on a remote server.

Scenario 4: Authorize valid SAML assertions

- Create an access control policy that handles client SOAP web service requests with the following conditions:
 - A SAML authentication assertion holds the requesting client identity
 - Accepts the claimed identity of the client if the digital signature of the SAML assertion is valid
 - The requested resource is defined as an attribute in the SAML assertion
 - Allows any authenticated client with a specific SAML attribute access to the web service operation

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-29. Scenario 4: Authorize valid SAML assertions

In this example, the request message contains a SAML authentication statement and a SAML attribute statement. The authentication statement claims that the current requester is verified during a previous processing step. The access control policy accepts this claim if and only if the digital signature that was used to sign the claim is valid.

An application-specific SAML attribute describes the resource that the client requests. The policy authorizes the request if the current requester is an authorized user.

Scenario 4: SAML authentication statement (1 of 2)

```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
  AssertionID="IDd600a593-4e13-44d9-829a-3055600c46ca"
  IssueInstant="2006-07-28T18:51:02Z"
    Issuer=http://training.ibm.com/security/
  MajorVersion="1" MinorVersion="1">
  <saml:Conditions NotBefore="2006-07-28T18:51:02Z"
    NotOnOrAfter="2006-07-28T18:54:02Z"/>
  <saml:AuthenticationStatement
    AuthenticationInstant="2006-07-28T18:51:02Z"
    AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:unspecified">
    <saml:Subject>
      <saml:NameIdentifier
        Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
        NameQualifier="http://address.training.ibm.com">
        admin
      </saml:NameIdentifier>
```

. . . (continued on next slide)

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-30. Scenario 4: SAML authentication statement (1 of 2)

This example is a SAML assertion that is generated in the post-processing step of an access control policy.

The **Conditions** element defines a window of time in which this statement is valid. This time limit reduces the likelihood of a replay attack.

Within the **AuthenticationStatement**, the **Subject** element describes the identity of the client through a **NameIdentifier** element.

This statement is a SAML V1.1 authentication statement. The syntax for a SAML V2.0 authentication statement is different from this structure. IBM DataPower supports all of SAML V1.0, V1.1, and V2.0 statements.

Scenario 4: SAML authentication statement (2 of 2)

. . . (continued from previous slide)

```
<saml:SubjectConfirmation>
  <saml:ConfirmationMethod>
    urn:oasis:names:tc:SAML:1.0:cm:sender-vouches
  </saml:ConfirmationMethod>
</saml:SubjectConfirmation>
</saml:Subject>
<saml:SubjectLocality IPAddress="127.0.0.1"/>
</saml:AuthenticationStatement>
</saml:Assertion>
```

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-31. Scenario 4: SAML authentication statement (2 of 2)

The `SubjectConfirmation` element describes which party backs up the claim. In this example, the message sender vouches for the validity of this claim.

It is a good practice to sign SAML assertions digitally to maintain the integrity of the claim.

Scenario 4: SAML attribute statement (1 of 2)

```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
... MajorVersion="1" MinorVersion="1">
  <saml:Conditions NotBefore="2006-07-28T18:51:02Z"
    NotOnOrAfter="2006-07-28T18:54:02Z"/>
  <saml:AttributeStatement>
    <saml:Subject>
      <saml:NameIdentifier
        Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
        NameQualifier="http://address.training.ibm.com">
        admin
      </saml:NameIdentifier>
    </saml:Subject>
```

. . . (continued on next slide)

Figure 1-32. Scenario 4: SAML attribute statement (1 of 2)

This example is a SAML attribute statement, holding application-specific information.

Similar to a SAML authentication statement, the `NameIdentifier` element describes the subject that added the attribute.

Scenario 4: SAML attribute statement (2 of 2)

. . . (continued from previous slide)

```
<saml:Attribute
  AttributeName="EastAddressSearch"
  AttributeNamespace="http://address.training.ibm.com">
  <saml:AttributeValue>
    Query
  </saml:AttributeValue>
</saml:Attribute>
</saml:AttributeStatement>
</saml:Assertion>
```

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-33. Scenario 4: SAML attribute statement (2 of 2)

The `Attribute` element describes application-specific information. For example, a SAML attribute element can encapsulate fields from an LDAP directory entry. The system can use this additional information about the subject to make an authorization decision.

Again, it is a good practice to sign SAML assertions digitally to maintain the integrity of the claim.

Scenario 4: Identify and authenticate the client

1. Create a AAA policy object on the DataPower gateway

2. Extract the client's identity by the **Name from SAML Authentication assertion** option

3. For the authentication method, select **Accept a SAML assertion with valid signature**
 - Specify the validation credential for the SAML signature
 - If blank, certificate validation is skipped

4. Leave the identity mapping method at **None**

The figure consists of two screenshots from the IBM DataPower configuration console.

The top screenshot shows the 'Identification Methods' section. It contains a list of options for identifying the client:

- ☐ Subject DN or SSL certificate from connection
- ☐ Name from SAML Attribute assertion
- ☒ Name from SAML Authentication assertion
- ☐ SAML Artifact
- ☐ Client IP address

The bottom screenshot shows the 'Define how to authenticate the user' section. It contains a list of authentication methods:

- ☐ Accept LTPA token
- ☒ Accept SAML assertion with valid signature
- ☐ Bind to LDAP server
- ☐ Validate SSL certificate from connection peer

 Below these options is a section for 'SAML Signature Validation Credentials'. It includes a dropdown menu showing 'oauthValCred', and two buttons: a plus sign (+) and a minus sign (-).

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-34. Scenario 4: Identify and authenticate the client

To verify the signature of the SAML assertion, the access control policy needs the validation credential. If the validation credentials field is blank, then the certificate is not validated. In either case, the signature is verified.

Scenario 4: Authorize access to resources

5. Select **Local name of request element** as the resource extraction method

- The name of the child element in the SOAP body of the request is the request element name

6. For the authorization method, **Use SAML attributes from authentication**

- Set the SAML attribute that matches type as **Any**

7. Click **SAML Attributes** from the authentication method page

Resource Identification Methods	
	<input type="checkbox"/> URL sent to back end
	<input type="checkbox"/> URL sent by client
	<input type="checkbox"/> URI of toplevel element in the message
	<input checked="" type="checkbox"/> Local name of request element
	<input type="checkbox"/> HTTP operation (GET/POST)
	<input type="checkbox"/> XPath expression
	*

Define how to authorize a request.

☐ AAA information file
☐ Generate SAML Authorization query
☒ Use SAML attributes from authentication
☐ Use XACML Authorization decision

☐ All values
☐ All
☐ Any value
☒ Any
☐ XPath

Type

SAML Attributes

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-35. Scenario 4: Authorize access to resources

When authorizing requests based on SAML attributes, you must specify one or more expected attributes in a separate page. The following slide describes how to enter in the list of expected SAML attributes.

Scenario 4: Match SAML attributes

8. On the **SAML Attributes** page, click **Add**
9. Declare the expected SAML attribute values within an SAML attribute statement

- The namespace URI and local name represent the qualified name for the SAML attribute
- The attribute value is application-specific; it can be used to represent the identity of the client or the name of a requested resource

Add a SAML Attribute

Namespace URI	<input type="text" value="i.ibm.com/datapower/FLY/BookingService"/>
Local name	<input type="text" value="BookingService"/>
Attribute value	<input type="text" value="Request"/>

SAML Attributes		
Namespace URI	Local name	Attribute value
http://www.ibm.com/datapower/FLY/BookingService	BookingService	Request

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-36. Scenario 4: Match SAML attributes

Access control policy by SAML information

- Identity extraction methods:
 - Name from SAML attribute assertion `<saml:Subject>` element
 - Name from SAML authentication assertion `<saml:Subject>` element
 - SAML browser artifact from the URL query string
- Authentication methods:
 - Accept a SAML assertion with a valid signature
 - Retrieve SAML assertions corresponding to a SAML browser artifact
 - Contact a SAML server for a SAML authentication statement
- Authorization methods:
 - Generate a SAML authorization query
 - Generate a SAML attribute query
- Postprocessing:
 - Generate a SAML V1.0, V1.1, or V2.0 assertion

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-37. Access control policy by SAML information

In addition to the previously covered scenarios, an access control policy can parse any token type and make a SAML authorization or attribute query to an external server. To avoid repeating security checks, the policy can generate a SAML assertion during the post processing stage.

Unit summary

- Describe the AAA framework within the DataPower Gateway
- Explain the purpose of each step in an access control policy
- Authenticate and authorize requests with:
 - WS-Security Username and binary security tokens
 - HTTP Authorization header claims
 - Security Assertion Markup Language (SAML) assertions

Review questions



1. True or False: To authenticate a client without using an external access control resource, you can compare the client's credentials against a custom DataPower AAA information file or validate the digital signature that is used to sign the credential.
2. True or False: If the Authenticate step fails, the Extract Resource step is not attempted.
3. True or False: The postprocessing step in an access control policy adds more information to the outgoing request message or transforms the message itself.

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-39. Review questions

Write your answers here:

- 1.
- 2.
- 3.

Review answers



1. True or False: To authenticate a client without using an external access control resource, you can compare the client's credentials against a custom DataPower AAA information file or validate the digital signature that is used to sign the credential.
The answer is True.
2. True or False: If the Authenticate step fails, the Extract Resource step is not attempted.
The answer is False. Even if the Authenticate step fails, the Extract Resource step occurs. In fact, although the authentication might fail, the Authorize step always occurs because all requests might be allowed.
3. True or False: The postprocessing step in an access control policy adds more information to the outgoing request message or transforms the message itself.
The answer is True. Extra tokens such as a SAML assertion or LTPA token can be added to the original message. The postprocessing step also supports using a stylesheet or GatewayScript for further processing of the message.

Exercise: Configuring authentication and authorization in a service

Authentication, authorization, and auditing (AAA)

© Copyright IBM Corporation 2017

Figure 1-41. Exercise: Configuring authentication and authorization in a service

Exercise objectives



- Configure a AAA action to enforce authentication and authorization policies that are in a AAA information file
- Configure a AAA action to enforce authentication and authorization policies that are in an LDAP server

Unit 2. OAuth overview and DataPower implementation

Estimated time

00:45

Overview

This unit introduces the OAuth 2.0 framework and its DataPower implementation.

How you will check your progress

- Checkpoint
- Hands-on exercise

References

IBM DataPower Gateway Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.5.0

Unit objectives

- Describe the OAuth framework
- Describe why OAuth is useful in security scenarios
- Describe the OAuth three-legged scenario
- Explain the role that a DataPower gateway performs in an OAuth framework
- Describe the OAuth configuration options on DataPower: the web token service, the AAA action, the OAuth client profile, and the OAuth client group

What is OAuth?

- OAuth defines a way for a client to access server resources *on behalf* of another party
- It provides a way for the user to authorize a third party to their server resources *without* sharing their credentials
- It *separates* the identity of the resource owner (user) from a third-party application that acts on behalf of the user
- It allows a user to grant *different levels of access* to different third-party applications, for a specified duration of time
- OAuth 2.0 authorization framework: <http://tools.ietf.org/html/rfc6749>

OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-2. What is OAuth?

The OAuth specification solves a specific problem: how to delegate access rights to a third-party client that is working on behalf of the user. Before OAuth, third-party applications asked and stored the user's user name and password within the application. This process is risky because the server cannot distinguish between the user and the third-party application. One analogy in the real world is to hand over your house keys to a cleaning service. You must have a high degree of trust in the client to give them complete access to your home.

With OAuth, the client does not use your credentials. Instead, an authorization service gives a temporary pass to the client, so it can perform a limited set of tasks in a fixed time period. As the user, you can tell the authorization service to revoke the temporary pass at any time.

Although OAuth is more complicated than handing over your credentials to the client, it is a safer mechanism that gives the user control over the third-party client's actions.

Delegated authorization example

- When you purchase a car, you receive a main key and a *valet* key.



- The valet can use your car with this key, with some restrictions:
 - Speed restriction
 - Distance restriction
 - Cannot alter some car functions, such as radio stations
 - Cannot open car storage areas
- When you allow the valet to borrow the valet key, you delegate access to certain features of the car to a third party.

OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

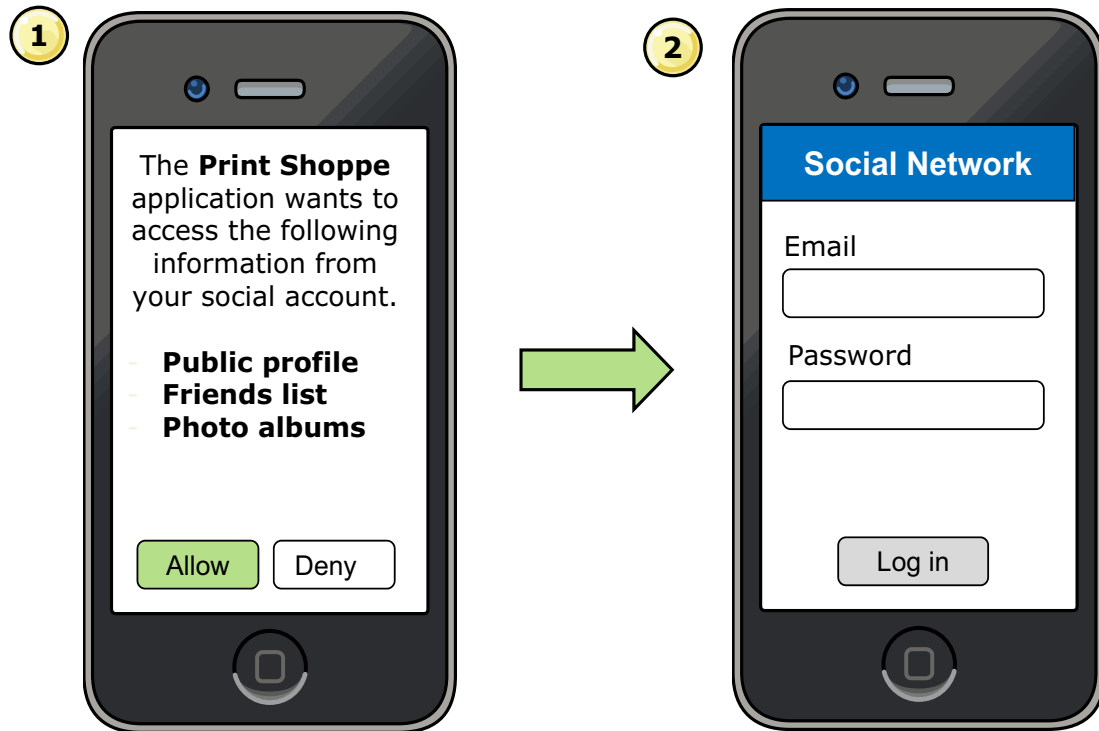
Figure 2-3. Delegated authorization example

When you purchase a car, you receive a main key and a valet key.

The valet can use your car with this key, with some restrictions. The car cannot exceed a certain speed. The car might not be able to travel as far as with the regular key. The valet cannot change the radio station settings. The valet cannot open car storage areas.

When you allow the valet to borrow the valet key, you delegate access to certain features of the car to a third party.

Example: Allow third-party access to social account



OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-4. Example: Allow third-party access to social account

Whenever you sign up for a web-based application or a mobile application, you create an account on the server with a user name and password. The process becomes tedious for the user when they sign up for dozens of applications.

Social networks, such as Facebook and Twitter, already link your identity to a user account. Therefore, many applications use your social network account to create an account.

There are three players in this scenario. You as the user; the third-party application as a client; and the social network as a web-based service. You want the third-party application to access some (but not all) of your information from the service. That is, you want the client to act on your behalf to access resources on the service.

In this example, the third-party application, the Print Shoppe, wants to access your online photo album from your social network account. The application opens a new page from the social network site. After you log in to the social network, the social network service grants an authorization token to the application. At no time does the third-party application see your user name or password on the social network.

Example: Third-party access to online photo album

- OAuth allows social network applications to share resources.



Alice is the **owner** of an online photo album. As a **resource owner**, Alice declares which applications can access her online photo album.



The Print Shoppe, a third-party **client application**, wants to access Alice's photos from an online service.



The social network provides an online photo album. This service acts as the **resource server**. It manages access to Alice's photos.



An **authorization server** verifies the identity of the client that wants to access Alice's photo album. This server issues a token or a code to access the photo album from the resource server.

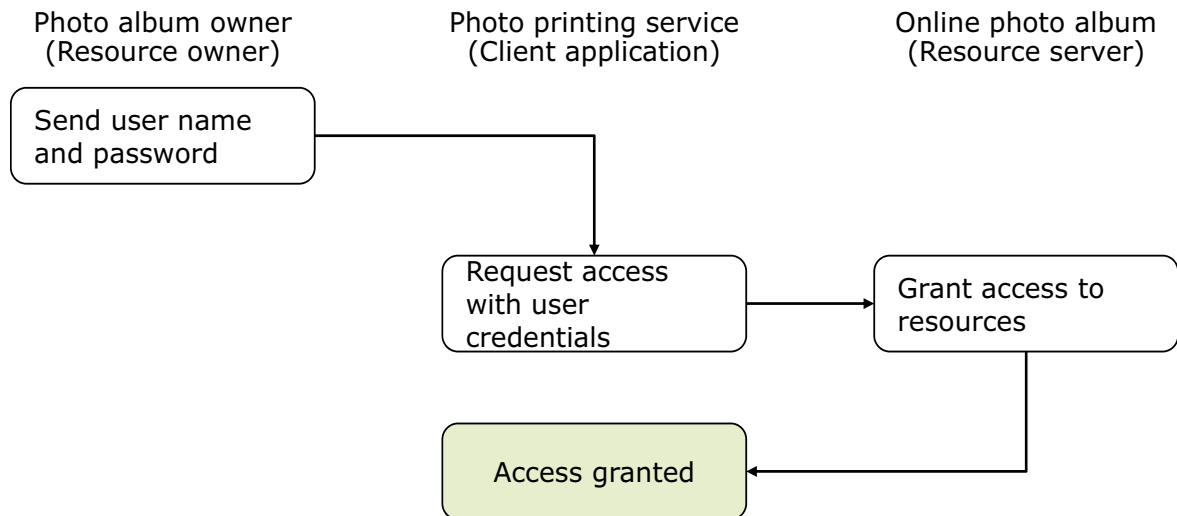
OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-5. Example: Third-party access to online photo album

Take a closer look at the three actors in the OAuth scenario. Alice is the **owner** of an online photo album. As the **user**, Alice wants to print her photos with a third-party photo printing service. The Print Shoppe is a **third-party client application** that wants to access Alice's photos from the online service. Last, the social network is a service that securely stores Alice's photos. This service also manages access to the photos from Alice and third-party applications that act on Alice's behalf.

Before OAuth: Sharing user passwords



- Issues with sharing passwords:
 - The service cannot distinguish between the user (resource owner) and the third-party application.
 - No method to revoke access for just the third-party application.

OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-6. Before OAuth: Sharing user passwords

Without OAuth, the user must give their user name and password to the third-party application. In turn, the third-party application sends these credentials while posing as the user. For convenience's sake, the application saves a copy of the user name and password.

There are several issues with this scenario. First, the service cannot distinguish between the owner of the resource, and the third-party application. To the service, it is the same user that is accessing the application. This practice is not safe; the user does not know what the application reads or modifies on the service. Second, there is no simple way to revoke access for one particular third-party application. The user must reset their password, which breaks access from all third-party applications.

OAuth Step 1: Resource owner requests access

- Step 1: Alice, as the resource owner, requests access to the online photo album (protected resource) for the printing service (OAuth Client)

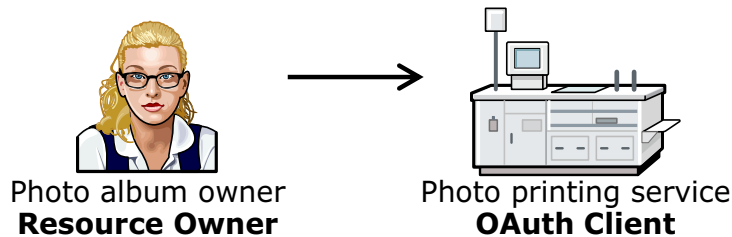
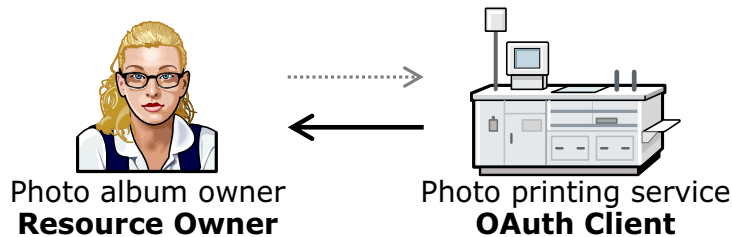


Figure 2-7. OAuth Step 1: Resource owner requests access

In this scenario, Alice is the owner of a photo album that is hosted on an online photo service. Alice wants to print a set of photos with a photo printing service. Alice is the resource owner, and the photo printing service is a third-party OAuth client application. Alice starts the process when she selects the "print from my photo album" option in the third-party application.

OAuth Step 2: OAuth client redirection to owner

- Step 2: The OAuth client sends the resource owner a redirection to the authorization server



OAuth overview and DataPower implementation

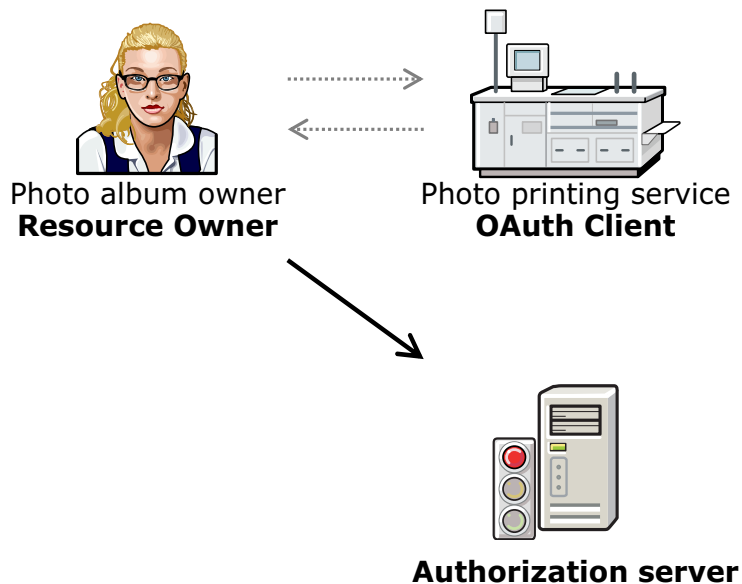
© Copyright IBM Corporation 2017

Figure 2-8. OAuth Step 2: OAuth client redirection to owner

In the second step, the third-party application requires the resource owner's authorization before it can access her online photo album. Instead of asking Alice directly for her user credentials, the third-party client application redirects Alice's request to an authorization server.

OAuth Step 3: Authenticate owner with authorization server

- Step 3: The resource owner authenticates against the authorization server



OAuth overview and DataPower implementation

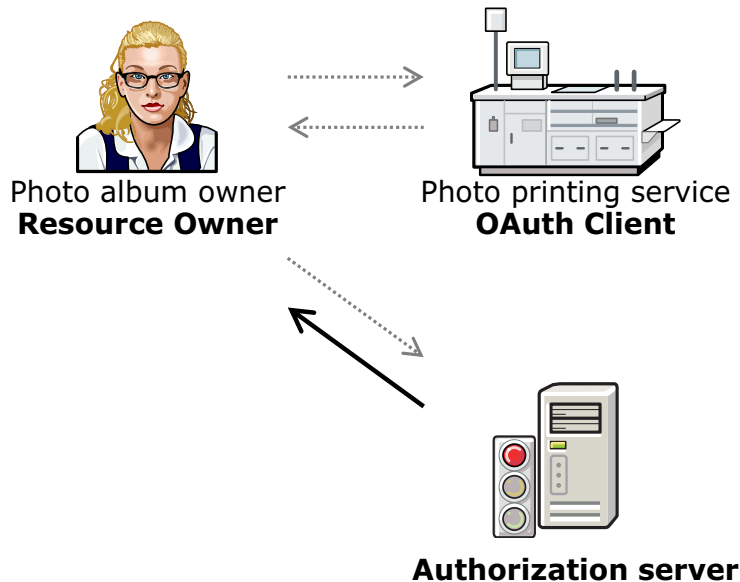
© Copyright IBM Corporation 2017

Figure 2-9. OAuth Step 3: Authenticate owner with authorization server

In the third step, the authorization server asks for Alice's user credentials to verify her identity.

OAuth Step 4: Ask resource owner to grant access to resources

- Step 4: The authorization server returns a web form to the resource owner to grant access



OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

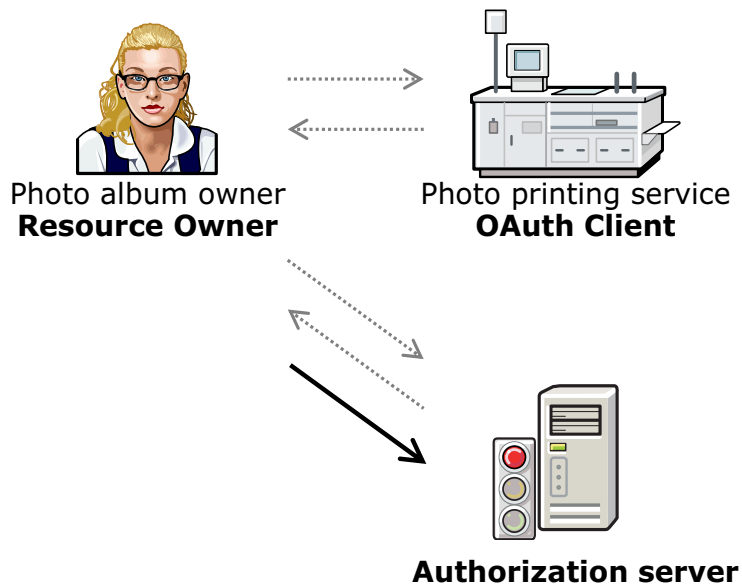
Figure 2-10. OAuth Step 4: Ask resource owner to grant access to resources

The authorization server returns a web form to ask Alice whether she grants the OAuth client access to her resources.

Notice that the ID and password interaction is between the authorization server and the resource owner. The OAuth client never sees that information.

OAuth Step 5: Resource owner grants client access to resources

- Step 5: The resource owner submits the form to allow or to deny access



OAuth overview and DataPower implementation

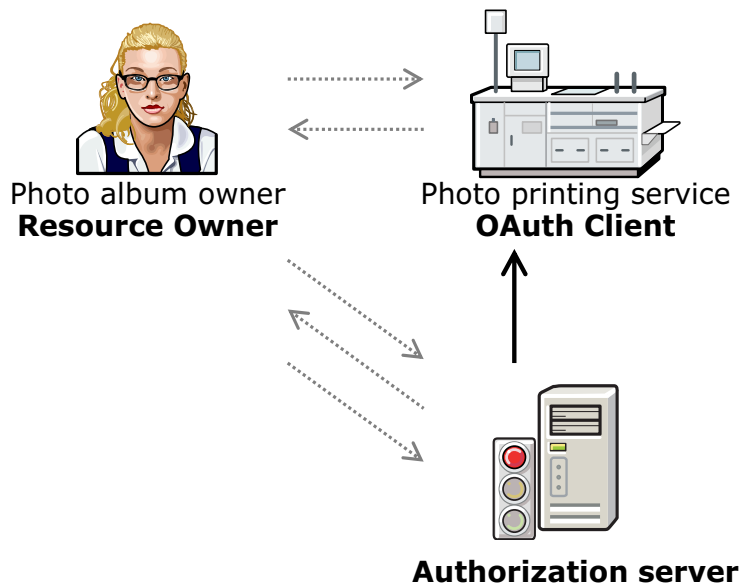
© Copyright IBM Corporation 2017

Figure 2-11. OAuth Step 5: Resource owner grants client access to resources

The resource owner, Alice, submits the web form to allow or deny access to her resources.

OAuth Step 6: Authorization server sends authorization grant code to client

- Step 6: If the resource owner allows access, the authorization server sends the OAuth client a redirection with the authorization grant code



OAuth overview and DataPower implementation

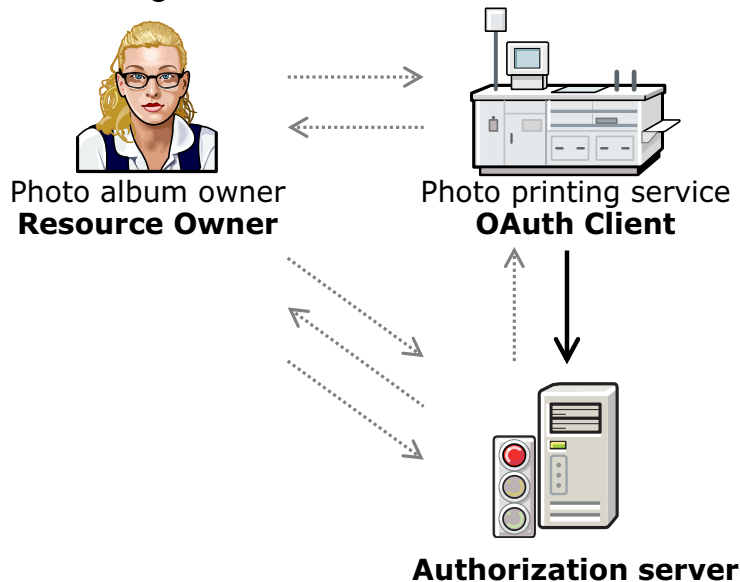
© Copyright IBM Corporation 2017

Figure 2-12. OAuth Step 6: Authorization server sends authorization grant code to client

The authorization server never transmits the resource owner's user name and password to the OAuth client. Instead, the server sends an authorization grant code: a code that specifies that the resource owner has authenticated to the authorization server and has granted access.

OAuth Step 7: Client requests access token from authorization server

- Step 7: To access the resource, the OAuth client sends the authorization grant code and other information to the authorization server to get an access token



OAuth overview and DataPower implementation

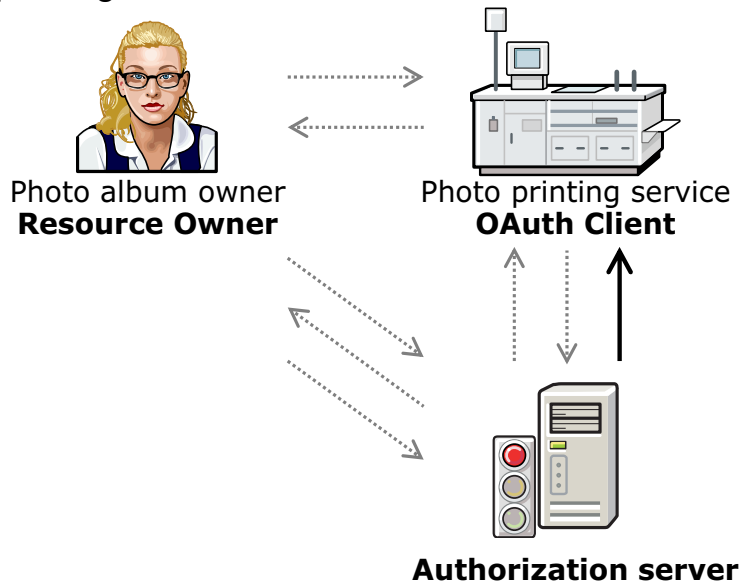
© Copyright IBM Corporation 2017

Figure 2-13. OAuth Step 7: Client requests access token from authorization server

The OAuth client sends three pieces of information to the authorization server: the authorization grant code, the client ID, and the client secret or client certificate. If the OAuth client is a public client, then it does not send the client secret or certificate.

OAuth Step 8: Authorization server sends authorization token to client

- Step 8: If the authorization server verifies the grant authorization information, it returns an access token to the OAuth client, the photo printing service



OAuth overview and DataPower implementation

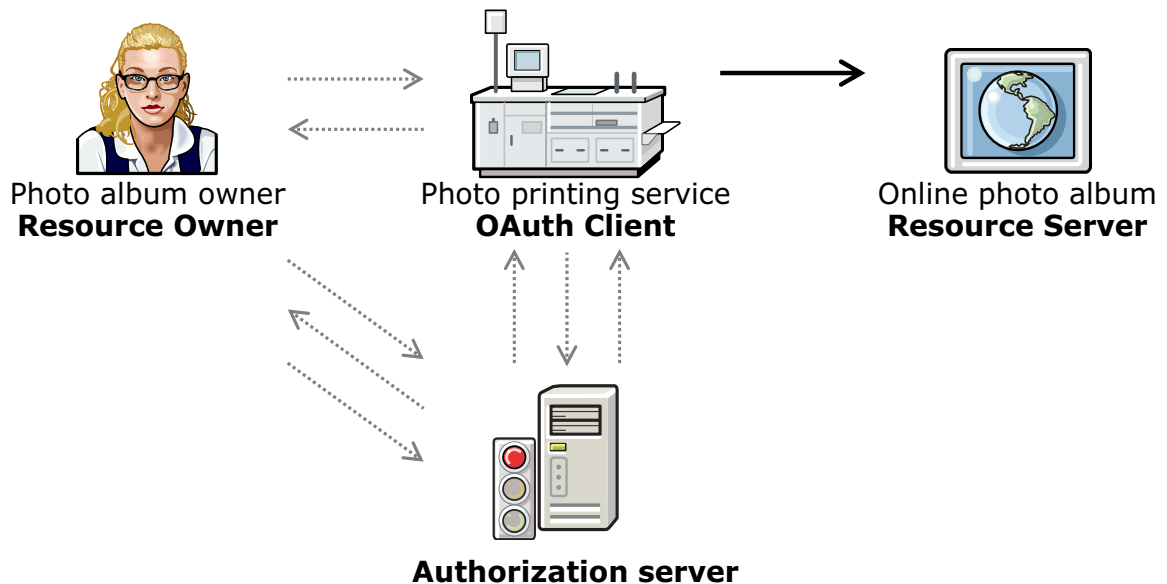
© Copyright IBM Corporation 2017

Figure 2-14. OAuth Step 8: Authorization server sends authorization token to client

Optionally, the authorization server can also return a refresh token. After the current access token expires, the OAuth client sends the refresh token to the authorization server to request another access token.

OAuth Step 9: OAuth client sends access token to resource server

- Step 9: The OAuth client sends the access token to the resource server



OAuth overview and DataPower implementation

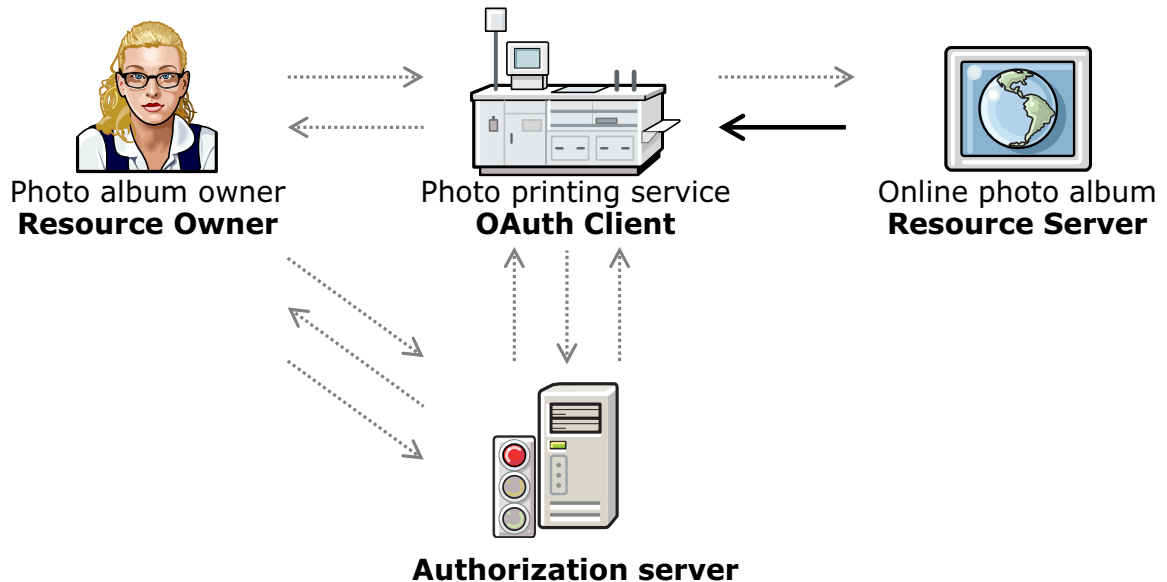
© Copyright IBM Corporation 2017

Figure 2-15. OAuth Step 9: OAuth client sends access token to resource server

It is possible that the authorization server and the resource server are the same server.

OAuth Step 10: Resource server grants access to OAuth client

- Step 10: If the access token is valid for the requested resource, the resource server allows the OAuth client to access the resource



OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-16. OAuth Step 10: Resource server grants access to OAuth client

Optionally, the authorization server can also return a refresh token. After the current access token expires, the OAuth client sends the refresh token to the authorization server to request another access token.

OAuth 2.0 protocol defines roles

Resource owner

An entity capable of granting access to a protected resource. When the resource owner is a person, it is referred to as a user.

Authorization server

The server that issues access tokens to the client after successfully authenticating the resource owner and obtaining authorization.

Client

An application that makes protected resource requests on behalf of the resource owner and with its authorization. The term "client" does not imply any particular implementation characteristics.

Resource server

The server that hosts the protected resources, capable of accepting and responding to protected resource requests by using access tokens.

Definitions from the OAuth 2.0 specification

[OAuth overview and DataPower implementation](#)

© Copyright IBM Corporation 2017

Figure 2-17. OAuth 2.0 protocol defines roles

These items are the role definitions that are directly from the specification.

OAuth 2.0 roles: Common implementation

Resource owner

An entity capable of granting access to a protected resource. When the resource owner is a person, it is referred to as a user.

Authorization and resource server

The server that provides the authorization activities and serves the protected resource

Client

An application that makes protected resource requests on behalf of the resource owner and with its authorization. The term “client” does not imply any particular implementation characteristics.

Typically called the three-legged scenario
Google and Yahoo OAuth APIs, for example, use this approach

[OAuth overview and DataPower implementation](#)

© Copyright IBM Corporation 2017

Figure 2-18. OAuth 2.0 roles: Common implementation

Although some implementations of OAuth combine the roles into what appears to be a single “server,” the roles and protocols are still the same.

OAuth 2.0 roles in the DataPower world (1 of 2)

Resource owner

Usually a user that interacts with the client and the various servers by using a user agent. A user agent can be a web browser, or an interface on a tablet or smartphone.

Authorization server

DataPower supplies a special service, the **Web Token Service**, which acts as an authorization server and token endpoint.

Client

Client code that is running on a web server, application that is running on a tablet or smartphone, client code that is running within a web browser. Varying levels of client credential confidentiality. Defined in DataPower as an **OAuth client profile** object.

Resource server

The resource server acts as a normal DataPower authentication server in front of the real back-end application, and can also perform other typical DataPower functions.

OAuth roles from a DataPower perspective

[OAuth overview and DataPower implementation](#)

© Copyright IBM Corporation 2017

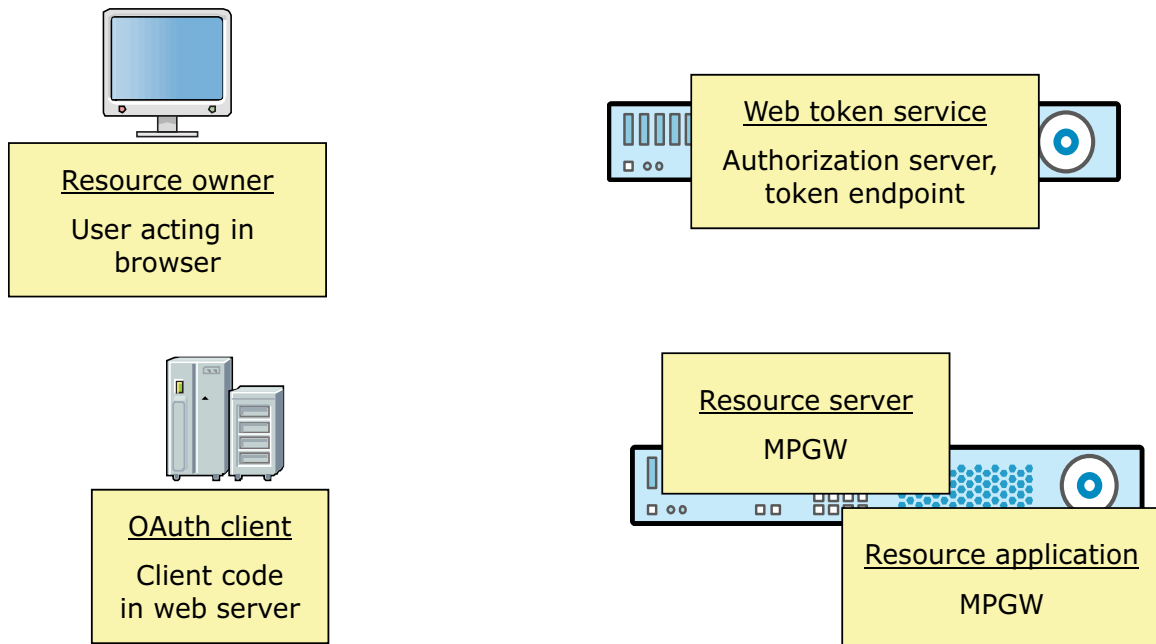
Figure 2-19. OAuth 2.0 roles in the DataPower world (1 of 2)

These names are the OAuth roles as they can be implemented in a DataPower environment.

DataPower does separate the authorization server function from the resource server function, contrary to many online systems that perform the two functions within the same system.

A web token service is defined as an authorization server and a token endpoint. The authorization behavior is as you expect. The token endpoint refers to the service's ability to supply an access token back to the client.

OAuth 2.0 roles in the DataPower world (2 of 2)



OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-20. OAuth 2.0 roles in the DataPower world (2 of 2)

The resource server is also considered as the enforcement point (EP) of the secured access to the actual back-end application.

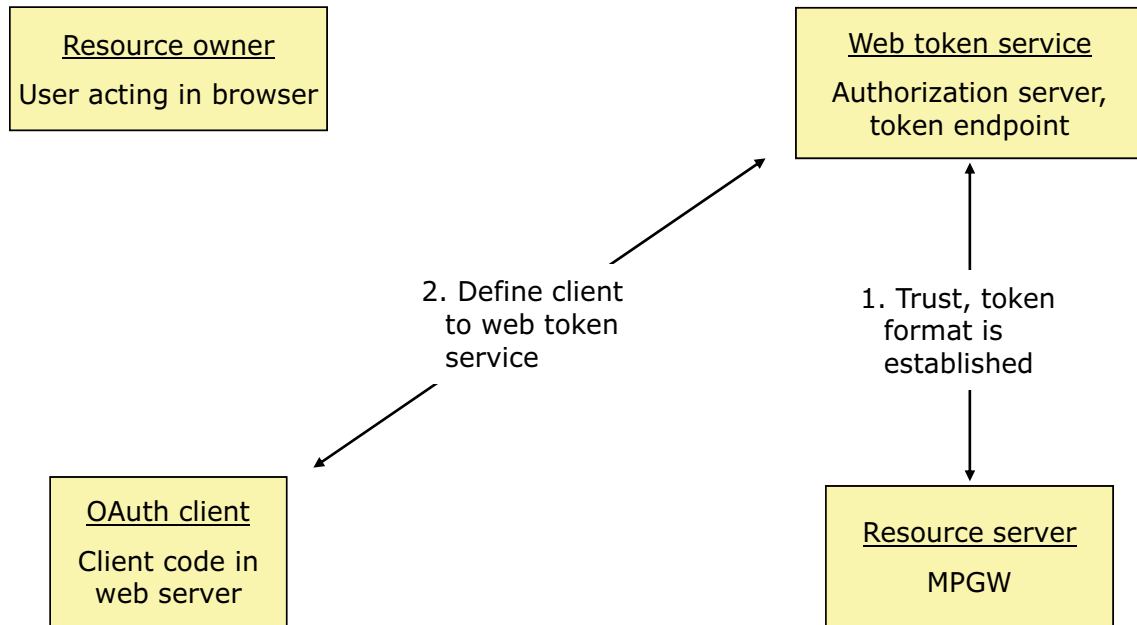
The resource application can be another service on the appliance, or an application service that is running on a server in the trusted domain.

DataPower also supports scenarios where the authorization server is **not** a DataPower service while the resource server is on DataPower, and the reverse situation.

Tivoli Federated Identity Manager also provides an OAuth authorization server implementation. A DataPower-based resource server can interact with the Tivoli Federated Identity Manager-based authorization server. The DataPower resource server sends a WS-Trust request to Tivoli Federated Identity Manager to have the access token validated.

Sample three-legged scenario in DataPower (1 of 4)

Activities before the first access



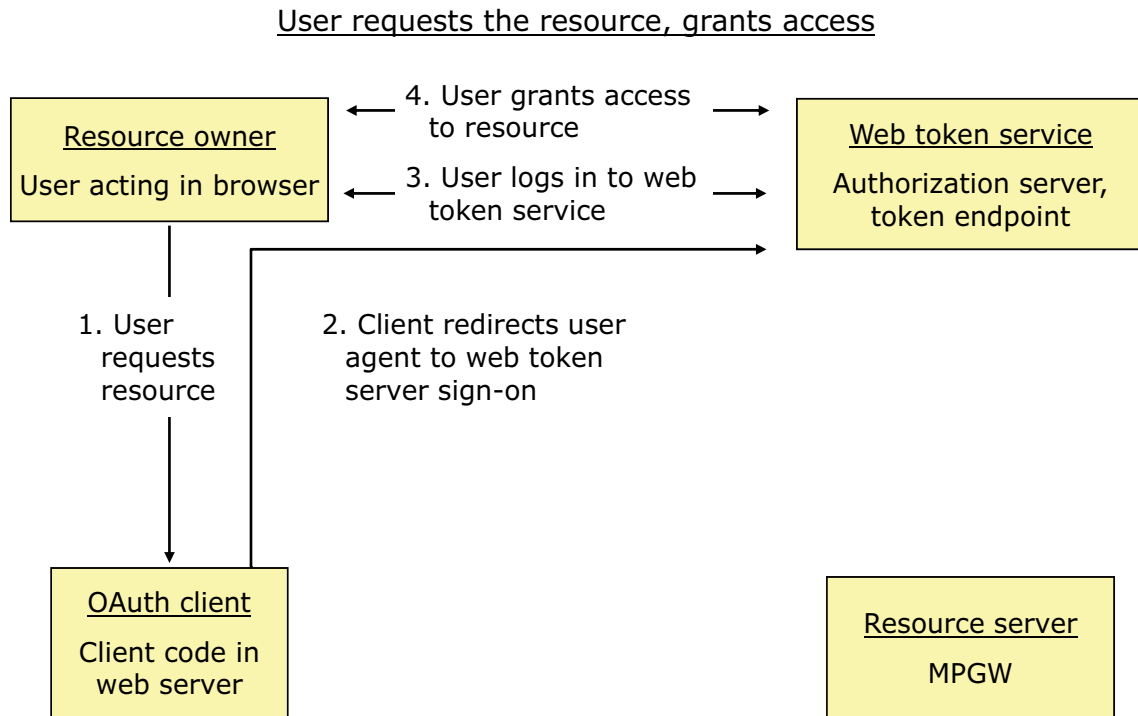
OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-21. Sample three-legged scenario in DataPower (1 of 4)

1. The OAuth 2.0 specification does not specify how the access token is formed or validated. DataPower defines its own implementation of the token format and validation, so both the web token service and the resource server use the same implementation.
2. The required parameters for the actual OAuth client are defined in an OAuth client profile object that the web token service and the resource server can access.

Sample three-legged scenario in DataPower (2 of 4)



OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

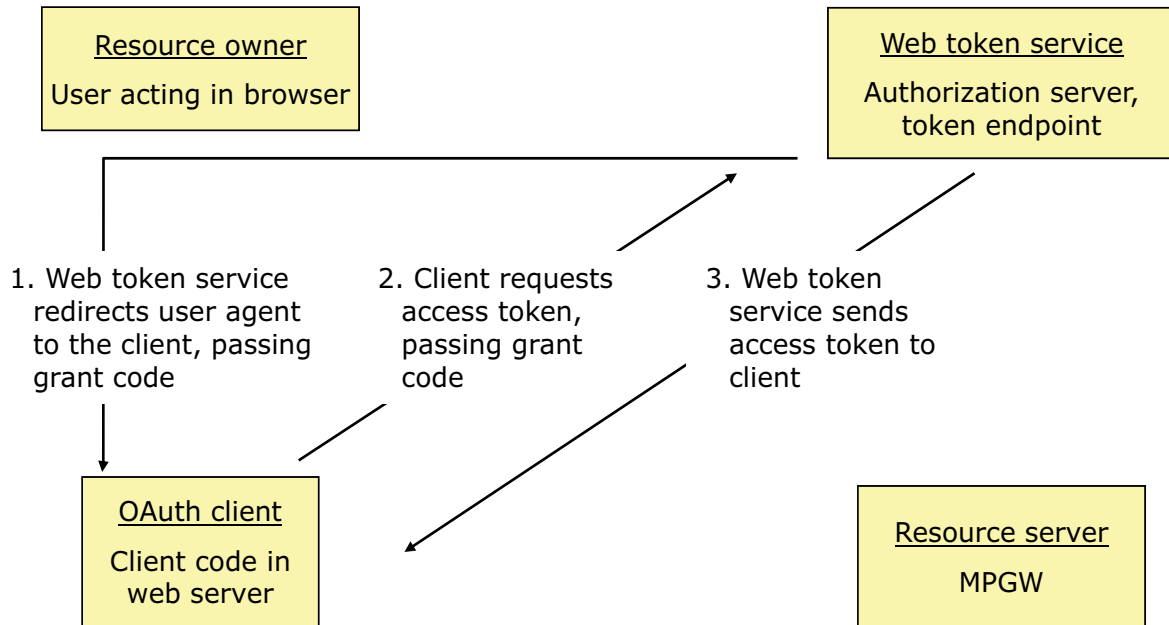
Figure 2-22. Sample three-legged scenario in DataPower (2 of 4)

This diagram is a sample flow of the grant type of authorization code. There are more grant types: implicit, resource owner password credentials, and client credentials. For more information, see the specification.

The user requests the resource from the client.

Sample three-legged scenario in DataPower (3 of 4)

Grant code is used to get access token



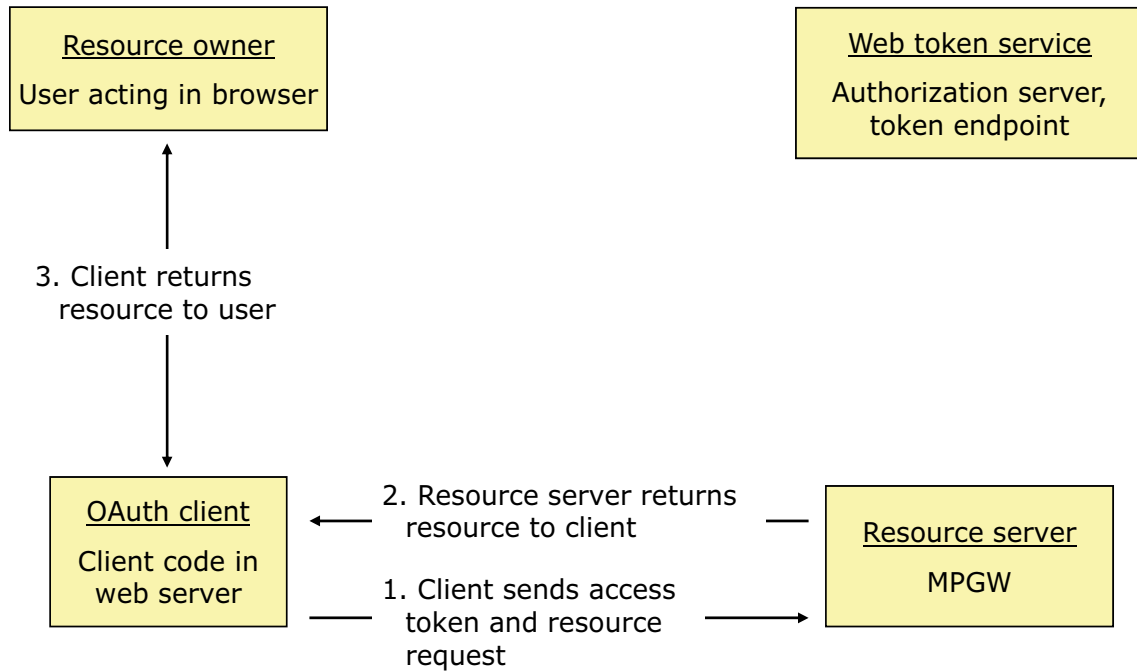
OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-23. Sample three-legged scenario in DataPower (3 of 4)

Sample three-legged scenario in DataPower (4 of 4)

Access token is used to get resource



OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-24. Sample three-legged scenario in DataPower (4 of 4)

Authorization request

The **OAuth client** issues a request for an **authorization grant** from the **authorization server**, on behalf of the user

The following parameters are sent as part of the URI string:

- **response_type**: A value of “code” identifies it as a request for an authorization grant
- **client_id**: The client identifier of the initiating OAuth client
 - This parameter is the client identifier that the authorization server knows each particular OAuth client by
- **redirect_uri**: A URL that refers to the OAuth client “entry point”
 - The authorization server uses this URL for an HTTP redirection on the response
- **scope**: The scope of the access request
- **state**: A value that the OAuth client can use to maintain state between the request and the callback
 - The authorization server includes this value when redirecting the user agent (browser) back to the client
 - The parameter should be used for preventing cross-site request forgery

OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-25. Authorization request

An example request from the lab exercise, from the OAuth client to the web token service:

```
https://172.16.78.12:7990/authz?response_type=code&client_id=FindBagOAuthClient&redirect_uri=http://172.16.80.23:4000/redirect&state=09e8fc0d1d2d&scope=findBag&name=1986
```

Authorization response

The **authorization server** responds with an **authorization grant code** to the **OAuth client**, by using an HTTP redirection to the user's browser (user agent)

The parameters:

- **code**: The authorization code that the authorization server generates
 - The authorization code must expire shortly after it is issued to mitigate the risk of leaks, and the client must not reuse it
- **state**: The “state” parameter that was present in the client authorization request

The parameters are returned in the URI string as part of the Location header in the redirection

Access token request

The **OAuth client** issues a request for an **access token** from the **authorization server**, on behalf of the OAuth client

- The authorization server is acting as a token endpoint

The parameters:

- **grant_type**: The value must be set to “authorization_code”
- **code**: The authorization code that was received from the authorization server
- **redirect_uri**: The “redirect_uri” parameter that was included in the authorization request
 - The authorization server requires that the values must be identical
- **client_id**: The client identifier of the initiating OAuth client
- **client_secret**: The client secret that is defined in the OAuth client

The parameters are sent in the HTTP request body

OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-27. Access token request

The client ID and the client secret can be sent in an HTTP Basic Authorization header. DataPower uses this approach.

Access token response

The **authorization server** responds with an **access token** to the **OAuth client**

- The authorization server is acting as a token endpoint

The parameters:

- **access_token**: The access token that the authorization server generates
- **token_type**: The type of the token
- **expires_in**: The lifetime in seconds of the access token
- **refresh_token**: (optional) The refresh token, which can be used to obtain new access tokens by using the same authorization grant code
- **scope**: The scope of the request

The response is returned in the HTTP request body as JSON data

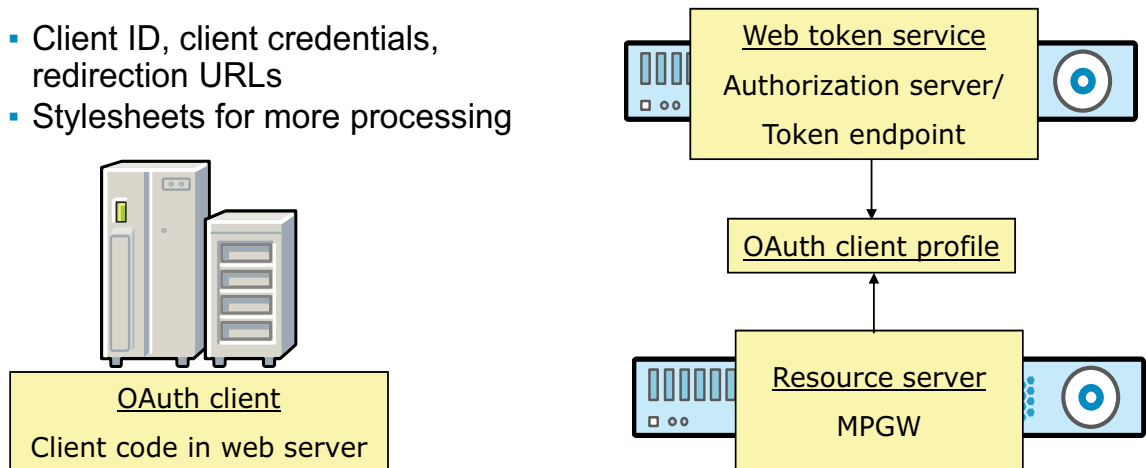
Resource request

- The **OAuth client** sends the request to the **resource server**, and includes the **access token**
- The specification does not explicitly indicate the parameters on the call, but it does indicate what must happen:
 - The resource server must validate the access token
 - The resource server must ensure that the token is not expired
 - The resource server must validate that the token scope covers the requested resource
- The methods that the resource server uses to validate the access token are beyond the scope of the specification. But they generally involve an interaction or coordination between the resource server and the authorization server.

OAuth client and the OAuth Client Profile object

- The AAA policies in the web token service and the resource server use the OAuth client profile to get details on a client that accesses the services:

- Client ID, client credentials, redirection URLs
- Stylesheets for more processing



- The actual OAuth client is running on some remote platform:
 - Web server
 - Smartphone

OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-30. OAuth client and the OAuth Client Profile object

DataPower OAuth objects: OAuth Client (1 of 4)

The screenshot shows the 'OAuth Client Profile' configuration page for an object named 'FindBagOAuthClient'. The interface includes a 'General' section with 'Administrative state' (set to 'enabled'), 'Comments', and 'Customized OAuth' (unchecked). Below this is the 'OAuth Role' section with 'Authorization and Token Endpoints' and 'Enforcement Point for Resource Server' checked. The 'Supported Type' section lists several grant types, with 'Authorization Code' checked. Three callout boxes provide additional context: the first points to the object name 'FindBagOAuthClient' stating it is used as the OAuth client ID; the second points to the 'Customized OAuth' checkbox stating it defines why the client is contacting the DataPower service; the third points to the 'Supported Type' section stating it specifies how this client can ask for a grant.

OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-31. DataPower OAuth objects: OAuth Client (1 of 4)

With the **customized OAuth** option, you can write your own OAuth client behavior in a stylesheet.

There are multiple options for OAuth 2.0 authorization grant types:

- With the authorization code, the authorization server sends back a custom redirect URI and an authorization code after it authenticates the resource owner. The authorization code prevents replay attacks. The client application opens the redirect URI with the authorization code to retrieve an access token for a resource.
- With the implicit grant type, the authorization server does not send back an authorization code. It sends back an access token after the resource owner authorizes the client application. This grant type is available for public clients only.
- With the resource owner password credentials grant type, the client application sends the user name and password for a user on the resource server. This grant type assumes a high level of trust between the client application and the resource server.
- With the client credentials grant type, the client application sends its own credentials when it accesses server resources under its own control, or to resources that are previously arranged with the resource server. This grant type is available to confidential client types only.
- With the JWT grant type, an encoded JWT is used to request an access token, rather than by using an authorization code. A JWT Bearer Token can be used to request an access token

when a client wishes to utilize an existing trust relationship, expressed through the semantics of the JWT, without a direct user-approval step at the authorization server. It also defines how a JWT can be used as a client authentication mechanism.

Authorization code and **Implicit grant** grant types are for three-legged OAuth flows. **Resource** owner password credential and **Client credential** are for two-legged OAuth flows.

The **validation grant** type allows a third party to contact the DataPower token endpoint to verify whether an access token that the appliance issues is valid. “Disable Validation Grant” disables that capability.

OpenID Connect allows an ID token to be generated.

DataPower OAuth objects: OAuth Client (2 of 4)

The screenshot shows the configuration for an OAuth Client in DataPower. The interface includes several fields and a checkbox, with annotations explaining their functions:

- Client Type:** Set to "Confidential". Annotation: "Confidential or public: Can client keep its credentials secret".
- Authentication Method:** Set to "Secret". Annotation: "How client authenticates: Client Secret or SSL certificate or JWT".
- Client Secret:** Set to "passw0rd". Annotation: "Text-based password".
- Customized Scope Check:** An unchecked checkbox.
- Scope:** Set to "/getAccountInfo". Annotation: "Regular expression to check the scope of the request".
- Shared Secret:** Set to "app-token-ssecret". Annotation: "Shared secret key to protect grant codes and access tokens".

OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-32. DataPower OAuth objects: OAuth Client (2 of 4)

With **Customized scope check**, you can use a stylesheet to do the scope checking, rather than using the **Scope** field entry.

DataPower OAuth objects: OAuth Client (3 of 4)

The screenshot shows the 'Authorization and Token Endpoints' section of the DataPower OAuth Client configuration. It includes a 'Redirect URI' field with a text input containing 'https://{1}\S+' and an 'add' button. Below this is an 'Authorization Form' dropdown menu with 'store:/// OAuth-Generate-HTML.xsl' selected. The 'Enforcement Point for Resource Server' section contains two checkboxes: 'Verify Client Credential' and 'Use Validation URL', both of which are currently unchecked. Annotations with arrows point to these elements: 'PCREs of valid redirection URIs that can be presented' points to the Redirect URI field; 'Stylesheet that generates the form to the resource owner' points to the Authorization Form dropdown; 'Option to verify the client' points to the Verify Client Credential checkbox; and 'Option to have a remote server validate the access token' points to the Use Validation URL checkbox.

OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-33. DataPower OAuth objects: OAuth Client (3 of 4)

With Verify client credential, you can verify the client identity along with using the access token.

Use validation URL is used when you want to use a remote server to validate the access token rather than using the DataPower resource server. This situation occurs when the authorization server is not a DataPower web token service.

DataPower OAuth objects: OAuth Client (4 of 4)

The screenshot shows the 'General' and 'Authorization and Token Endpoints' tabs of the DataPower OAuth Client configuration. Annotations point to specific fields:

- Caching mechanism:** Points to the 'Caching' dropdown menu, which is set to 'Replay Only'.
- Stylesheet/GatewayScript that executes after a successful OAuth request:** Points to the 'Additional OAuth Process' field, which contains 'local:///add-processing.xsl'.
- Value if not specified in request:** Points to the 'Default Scope' text input field.
- Expiration time in seconds:** Points to the 'Authorization Grant Lifetime' and 'Access Token Lifetime' numeric input fields, which are set to 300 and 3600 respectively.
- Option to use a stylesheet/GatewayScript to extract more resource owner information:** Points to the 'Custom Resource Owner Handling' checkbox, which is currently unchecked.

OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-34. DataPower OAuth objects: OAuth Client (4 of 4)

This slide covers options that are on the **Advanced** tab.

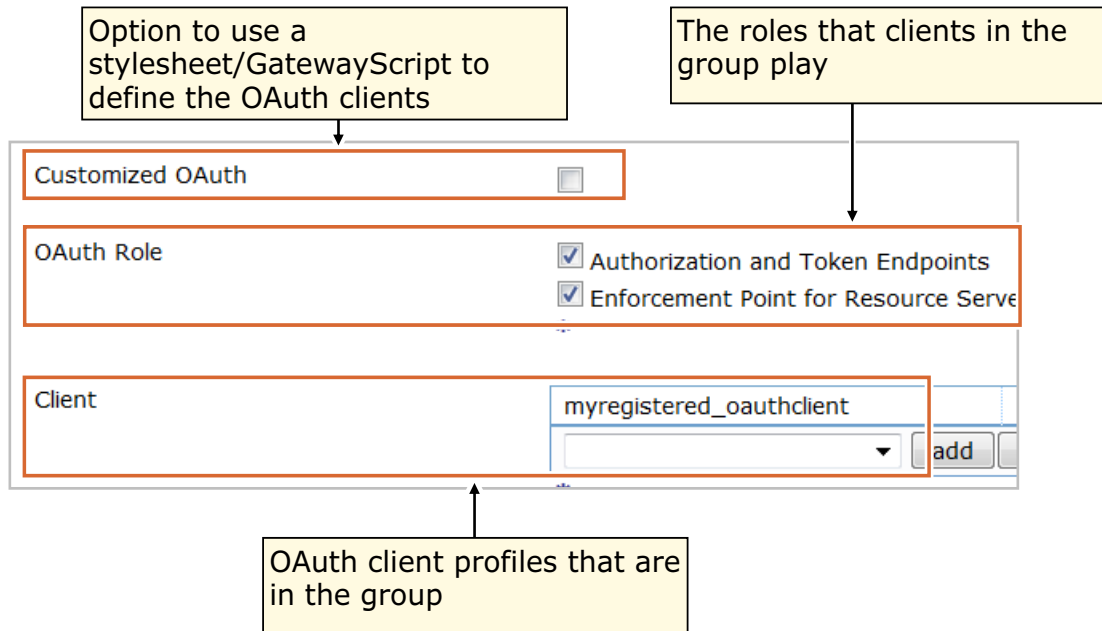
Caching specifies the caching mechanism to be used:

- **Replay Only:** Uses replay cache to prevent replay attacks
- **Token Cache:** Uses system memory to support revocation
- **Custom:** Uses a stylesheet that defines how to handle revocation
- **Distributed Cache:** Uses storage associated with the Quota Enforcement server

Other options for adding HTTP headers for resource owner, client ID, scope, and customized information are not shown in this screen capture.

DataPower OAuth objects: OAuth Client Group

- The AAA policies in the web token service and the resource server refer to a *group* of OAuth client profiles, rather than individual ones



OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-35. DataPower OAuth objects: OAuth Client Group

DataPower OAuth objects: Web Token Service

- The web token service is a specialized loopback service that acts as an authorization and token endpoint for OAuth
- Other tabs and fields allow specification of other service parameters, such as timeouts, request type, HTTP version, and other parameters

XML manager: default

Service priority: Normal

Endpoints

Local address	Port	SSL	SSL proxy profile (deprecated)	Credential Character Set	Security type
172.16.78.12	7990	on		Protocol	Se

Processing Policy: myWTS

OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-36. DataPower OAuth objects: Web Token Service

You can create your own DataPower service to act as the authorization server and token endpoint. It is much easier to use the web token service type that DataPower provides.

The **Client credential set** specifies the character encoding of the basic authentication values. "Protocol" indicates that the encoding is derived from what is specified in the HTTP protocol.

AAA policy: Key to OAuth behavior

The **AAA policy** within the processing policy of the web token service or the resource server controls the OAuth support

OAuth overview and DataPower implementation

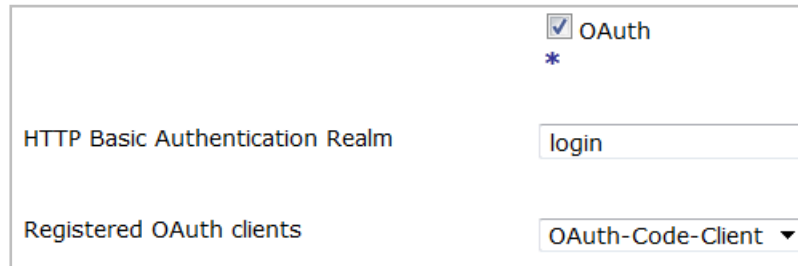
© Copyright IBM Corporation 2017

Figure 2-37. AAA policy: Key to OAuth behavior

The OAuth Client Profiles and OAuth Client Groups have no effect until they are referenced from a AAA policy.

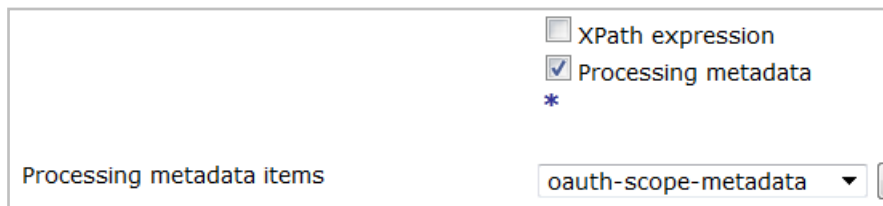
AAA policy for the web token service

- Along with the regular Identity extraction selection, you must also select **OAuth** and identify an **OAuth client group**



☒ OAuth *
 HTTP Basic Authentication Realm: login
 Registered OAuth clients: OAuth-Code-Client ▼

- In the Resource extraction phase, you select **Processing metadata**, and specify the **oauth-scope-metadata**



☐ XPath expression
☒ Processing metadata *
 Processing metadata items: oauth-scope-metadata ▼

OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

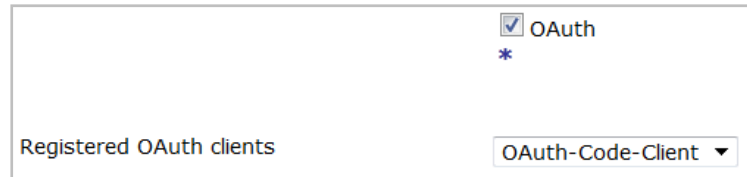
Figure 2-38. AAA policy for the web token service

In the top screen capture, “HTTP authentication header” is also selected. That choice is why the authentication realm is identified.

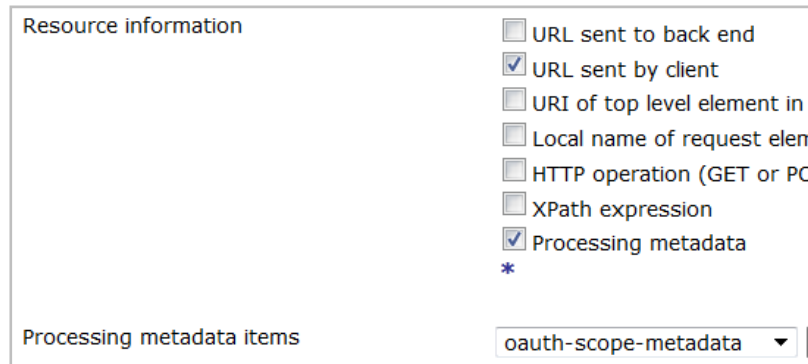
The oauth-scope-metadata contains the scope of the request.

AAA policy for the resource server (1 of 2)

- In Identity extraction, you must select **OAuth** and identify an **OAuth client group**



- In the Resource extraction phase, you select **URL sent by client** and **Processing metadata**, and specify the **oauth-scope-metadata**



OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-39. AAA policy for the resource server (1 of 2)

For V6.0 and later, the firmware verifies that the scope agreed to in the access token matches the actual requested scope (resource extraction phase).

AAA policy for the resource server (2 of 2)

- In the Map resources phase, select a **Method** of **Custom**, and specify the **stylesheet/GatewayScript** to perform the custom mapping
 - Required for pre-V6.0 firmware to compare the authorized scope to the actual requested scope

Method	Custom ▼ *
Custom URL	local:///accesstoken-check-resource

Unit summary

- Describe the OAuth framework
- Describe why OAuth is useful in security scenarios
- Describe the OAuth three-legged scenario
- Explain the role that a DataPower gateway performs in an OAuth framework
- Describe the OAuth configuration options on DataPower: the web token service, the AAA action, the OAuth client profile, and the OAuth client group

Review questions



1. True or False: With OAuth, resource owners allow third-party access to the resource without sharing their credentials.
2. True or False: Three-legged OAuth is the traffic and data pattern that OAuth is designed to solve.
3. DataPower implementation of OAuth includes (select 3):
 - A. Web Token Service
 - B. One-legged authentication
 - C. AAA action
 - D. SSL
 - E. OAuth Client Profile and OAuth Client Group
4. True or False: OAuth configuration on DataPower does not allow for the use of custom stylesheets.

OAuth overview and DataPower implementation

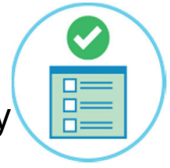
© Copyright IBM Corporation 2017

Figure 2-42. Review questions

Write your answers here:

- 1.
- 2.
- 3.
- 4.

Review answers



1. True or False: With OAuth, resource owners allow third-party access to the resource without sharing their credentials.
The answer is True.
2. True or False: Three-legged OAuth is the traffic and data pattern that OAuth is designed to solve.
The answer is True.
3. DataPower implementation of OAuth includes (select 3):
 - A. Web Token Service
 - B. One-legged authentication
 - C. AAA action
 - D. SSL
 - E. OAuth Client Profile and OAuth Client GroupThe answer is A, C, and E.
4. True or False: OAuth configuration on DataPower does not allow for the use of custom stylesheets.
The answer is False. OAuth configuration on DataPower *does* allow for the use of custom stylesheets.

Exercise: Defining a three-legged OAuth scenario that uses DataPower services

OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-44. Exercise: Defining a three-legged OAuth scenario that uses DataPower services

Exercise objectives



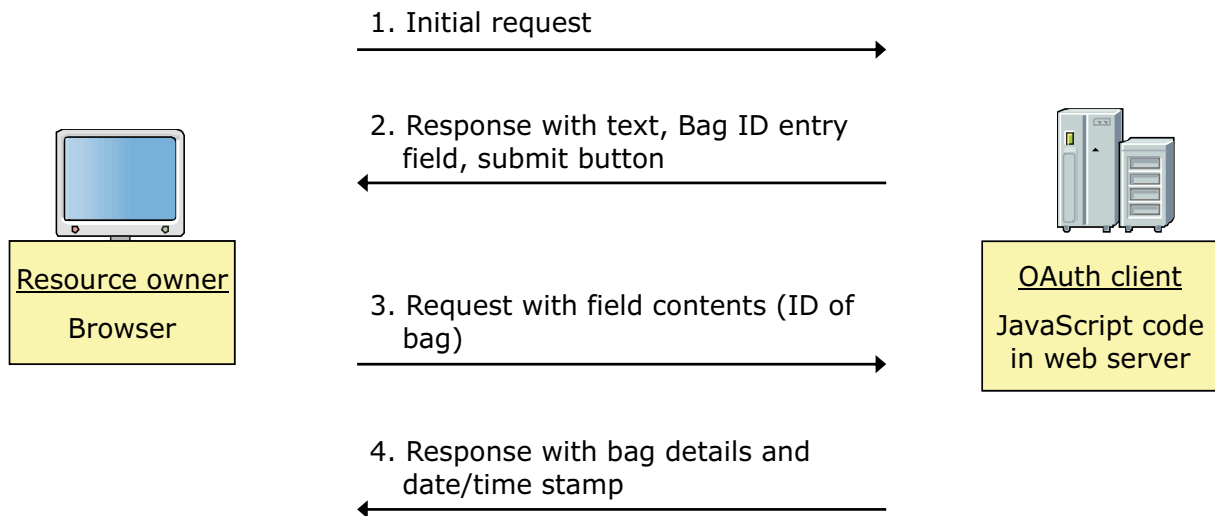
After completing this exercise, you should be able to:

- Define an OAuth Client Profile and an OAuth Client Group object
- Create a AAA policy to support the OAuth protocol
- Configure a DataPower web token service
- Configure a DataPower implementation of an OAuth resource server

Exercise overview: User interaction

Application function:

- User enters the ID of a bag to request its status
- Application responds with the current details on the bag, and the time stamp of the request

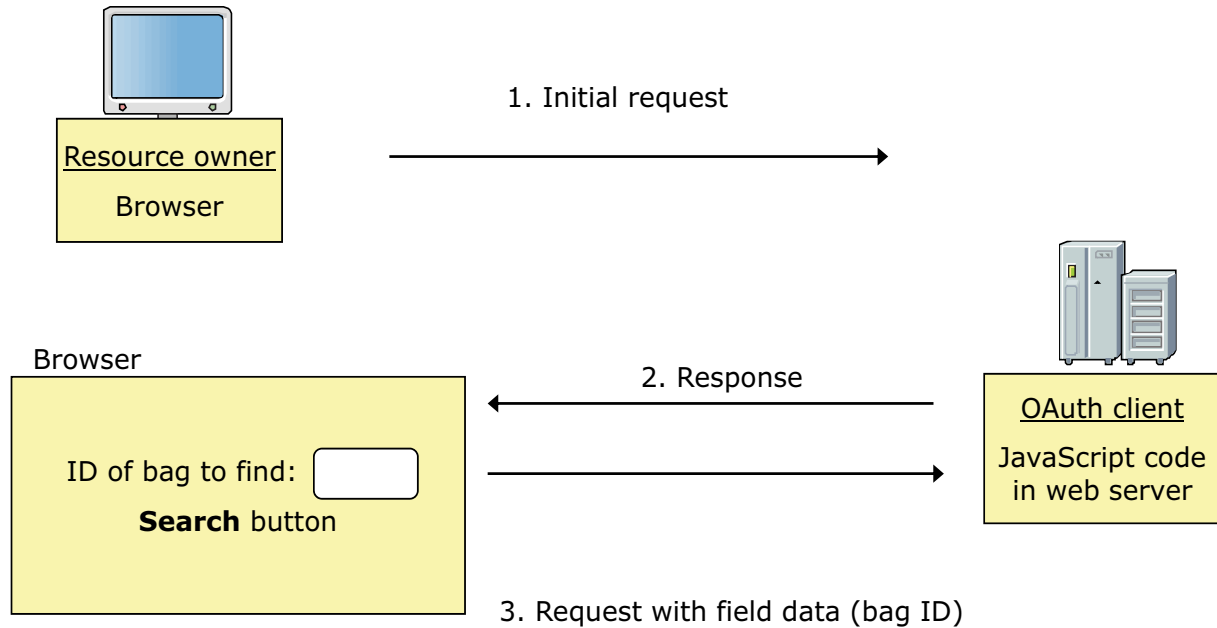


OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-46. Exercise overview: User interaction

Exercise overview: OAuth interaction (1 of 4)



OAuth overview and DataPower implementation

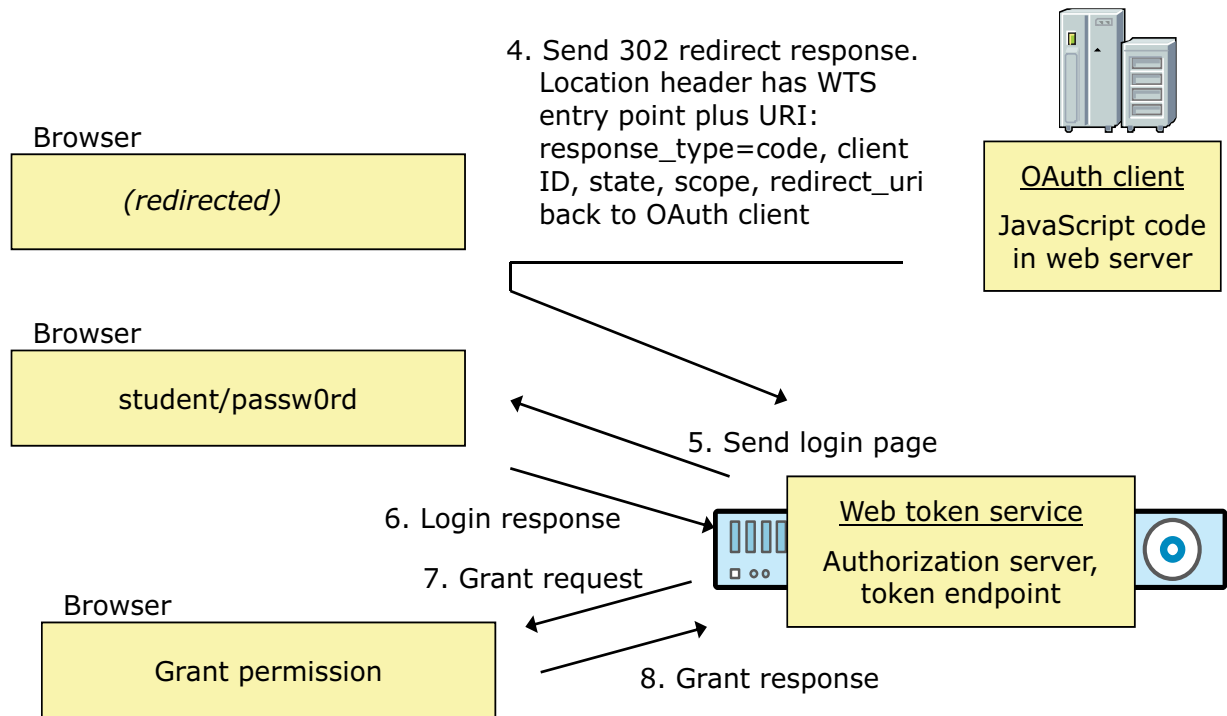
© Copyright IBM Corporation 2017

Figure 2-47. Exercise overview: OAuth interaction (1 of 4)

The OAuth client knows:

- Client ID
- Client secret
- Scope
- State
- Web token service URL
- Resource server URL
- Its own URL

Exercise overview: OAuth interaction (2 of 4)



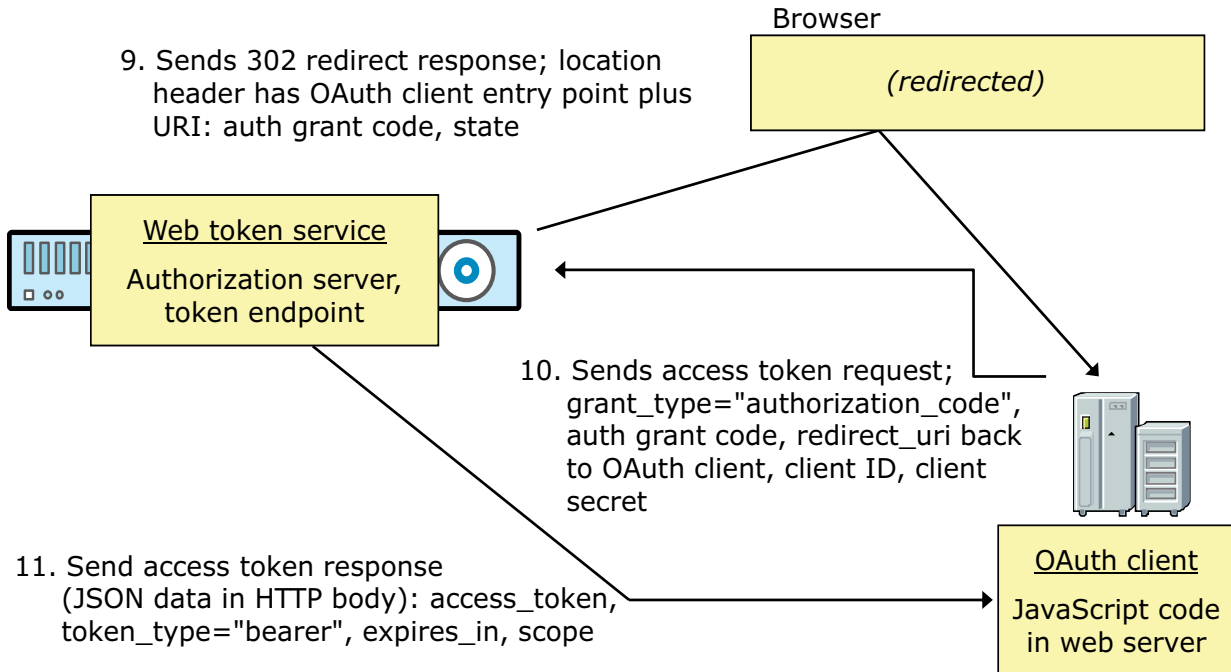
OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-48. Exercise overview: OAuth interaction (2 of 4)

Notice that the OAuth client is **not** involved in the login and grant interactions. The OAuth client never sees the ID and password of the user.

Exercise overview: OAuth interaction (3 of 4)



OAuth overview and DataPower implementation

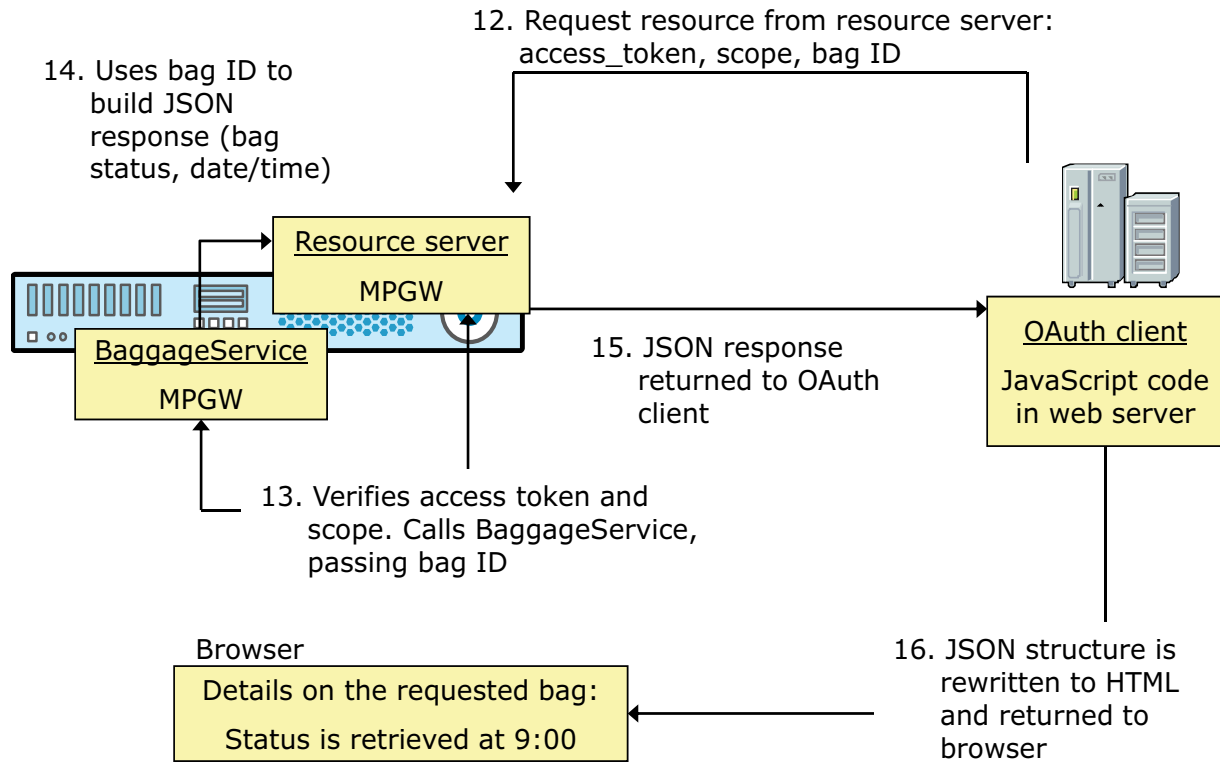
© Copyright IBM Corporation 2017

Figure 2-49. Exercise overview: OAuth interaction (3 of 4)

The user granted access permission to the OAuth client. Now the web token service is verifying the OAuth client before giving it an access token.

Notice that the browser does not see the access token.

Exercise overview: OAuth interaction (4 of 4)



OAuth overview and DataPower implementation

© Copyright IBM Corporation 2017

Figure 2-50. Exercise overview: OAuth interaction (4 of 4)

Unit 3. Social Login support in DataPower

Estimated time

00:45

Overview

This unit covers how DataPower supports Social Login.

How you will check your progress

- Checkpoint
- Hands-on exercise

References

IBM DataPower Gateway Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.5.0

Unit objectives

- Define Social Login
- Describe how to configure Social Login in DataPower
- Configure a Social Login Policy object
- Configure a JWT Generator and a JWT Validator object
- Describe OpenID Connect
- Configure an OpenID Connect client in DataPower

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-1. Unit objectives

Social Login and DataPower

- Social Login is authentication and authorization technique where:
 - A well-known social networking website (such as Google, Twitter, and Facebook) is used to authenticate and authorize a user to a third-party site
 - The third-party site does not see the login credential (userid and password) of the user
 - The third-party site does not need to maintain its own authentication and authorization server
 - The user does not need to create, remember, and maintain a separate login credential for the third-party site
- DataPower has built-in support for social login
 - Social login support can be added to new and existing services
 - Changes are primarily in the AAA Policy object and supporting objects
 - Social Login Policy, JWT Generator, JWT Validator objects
 - DataPower handles the protocol messages and browser redirects
 - Version 7.5.1 built-in support for social login is for social networking websites that implement OpenID Connect protocol (includes Google)
 - Facebook does not currently implement OIDC

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-2. Social Login and DataPower

What is OpenID Connect (OIDC)?

- OIDC enables clients to:
 - Verify the identity of the user based on the authentication that is performed by an authorization server
 - User credentials are not visible to the client (authorization code flow)
 - Obtain basic profile information about the user in an interoperable and REST-like manner
- OAuth provides only authorization, OIDC adds authentication
 - When using OAuth only, the resource server does not know the verified identity of the user
- OIDC works on top of the OAuth 2.0 framework
 - Adds protocol flows, token, endpoint, scopes
- OpenID Connect authentication layer specification:
http://openid.net/specs/openid-connect-core-1_0.html

OIDC terms

- End-User: OAuth resource owner
- OpenID Provider (OP): OAuth authorization server
- Relying Party (RP): OAuth client. It relies on the authorization server to verify the end-user's identity
- Claim: Piece of information asserted about an Entity (end-user)
 - Name, address, email, custom, and more
- ID Token: JSON Web Token (JWT) that contains Claims about the authentication event. It might contain other Claims
- UserInfo Endpoint: Endpoint on the OP that returns authorized information (claims) about the end-user
 - Additional endpoint to the OAuth authorization and token endpoints

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-4. OIDC terms

ID token

- Is a security token that contains claims about the authentication event of an end-user by an authorization server when using a client
 - Can potentially contain other requested claims
- Format is a signed JSON Web Token (JWT)
 - Optionally encrypted
- The ID token relates to authentication and identity similar to the way the access token relates to authorization

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-5. ID token

ID token: Required claims

Member	Description
iss	Issuer of the ID token. HTTPS URL to the issuer's endpoint
sub	Identifier of the verified subject. Locally unique case-sensitive string that is used by the client
aud	Audience that this ID token is intended for. Must contain the client_ID string of the relying party. May contain an array of strings for other audiences
exp	Expiration time of ID token. JSON number of seconds since January 1, 1970
iat	Time at which the ID token was issued. . JSON number of seconds since January 1, 1970

- More optional claims are defined, such as authorization time (auth_time) and a nonce string

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-6. ID token: Required claims

ID token: Example

```
{
  "iss": "https://server.example.com",
  "sub": "24400320",
  "aud": "s6BhdRkqt3",
  "nonce": "n-0S6_WzA2Mj",
  "exp": 1311281970,
  "iat": 1311280970,
}
```

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-7. ID token: Example

This example is from the OIDC specification.

Getting more information on the end-user

- More claims can be retrieved from the OP
 - A standard set is defined
 - Custom claims can be added
- The claims can be requested during the authorization phase
 - Additional “scope” parameters indicate the request for other claims
 - OIDC extends the use of the OAuth2 scope parameter to request claims
 - Defined “scope” values are: “email”, “address”, “phone”, “profile” (names, gender, birthdate, more)
 - Example: “scope=openid email” causes the email value to be returned in the ID token
- Claims can be requested in a subsequent call to the UserInfo endpoint
 - UserInfo-requested claims are returned in a JSON object
 - Can be signed and encrypted

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-8. Getting more information on the end-user

The “profile” scope returns the following standard claims: name, family_name, given_name, middle_name, nickname, preferred_username, profile, picture, website, gender, birthdate, zoneinfo, locale, and updated_at.

The OpenID Provider may or may not support individual claim requests.

Some of the standard claims

Member	Description
sub	Identifier of the verified subject. Locally unique case-sensitive string that is used by the client
name	End-user's full name in displayable form
given_name	Given name(s) or first name(s) of the end-user
family_name	Surname(s) or last name(s) of the end-user
email	End-user's preferred email address
address	End-user's preferred postal address. The value of the address member is a JSON structure containing the address subfields

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-9. Some of the standard claims

The full list of standard claims is: sub, name, given_name, family_name, middle_name, nickname, preferred_username, profile, picture, website, email, email_verified, gender, birthdate, zoneinfo, locale, phone_number, phone_number_verified, address, updated_at.

The address claim is a JSON object composed of the following claims: formatted, street_address, locality, region, postal_code, country.

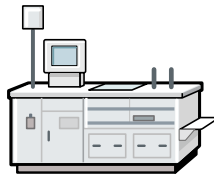
The given_name and family_name members can each contain multiple name strings.

Example flow for an end-user that must be authenticated

- This example shows the Authorization Code Flow
 - Other flows are Implicit Flow and Hybrid Flow
 - Similar to the Authorization Code Flow in OAuth 2.0
- The OP returns an access token and ID token from the token endpoint
- The OP authenticates the end-user without the RP seeing the end-user credentials
- The OP verifies the Client credentials



Alice is the **end-user** that needs to access a Relying Party to perform some function. Uses a browser.



The Print Shoppe, a **Relying Party** (RP), needs to authenticate the end-user before performing the requested function (OIDC client)



An **OpenID Provider** (OP) verifies the identity of the client and issues an ID token and access token. It can also provide additional claims for the end-user.

Social Login support in DataPower

© Copyright IBM Corporation 2017

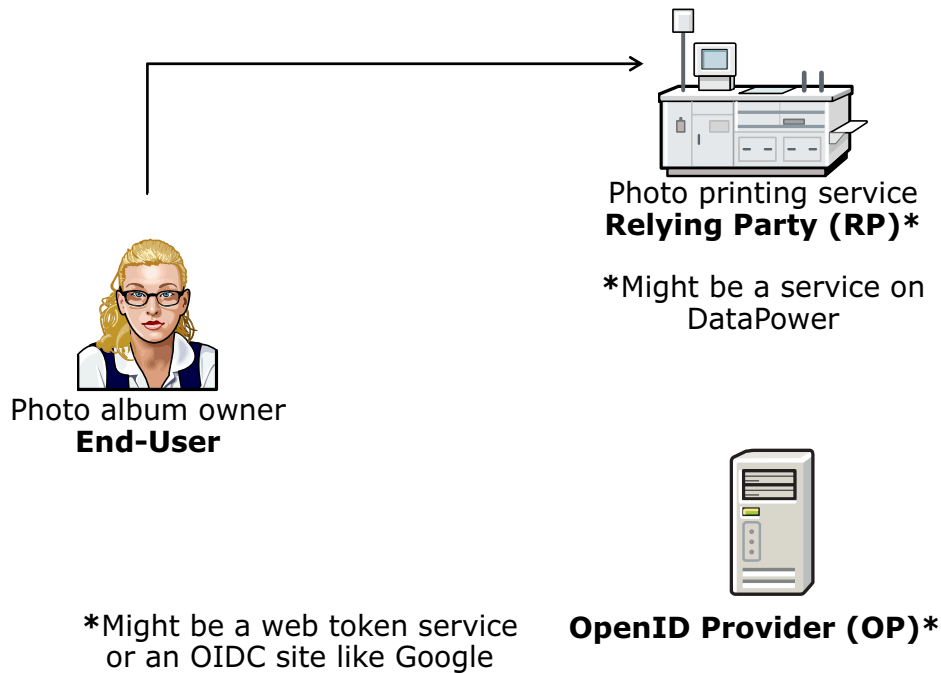
Figure 3-10. Example flow for an end-user that must be authenticated

The Authorization Code Flow returns an Authorization Code to the Relying Party, which can then exchange it for an ID Token and an Access Token directly. All tokens are returned from the Token Endpoint.

This provides the benefit of not exposing any tokens to the User Agent (typically a browser). The OpenID Provider can also authenticate the Relying Party before exchanging the authorization code for an access token. The Authorization Code Flow is suitable for Relying Parties that can securely maintain a “client secret” between themselves and the OpenID Provider.

OIDC Step 1: End-user initiates request

- Step 1: Alice, as the end-user, initiates a request to the RP that requires the RP to authenticate the end-user



Social Login support in DataPower

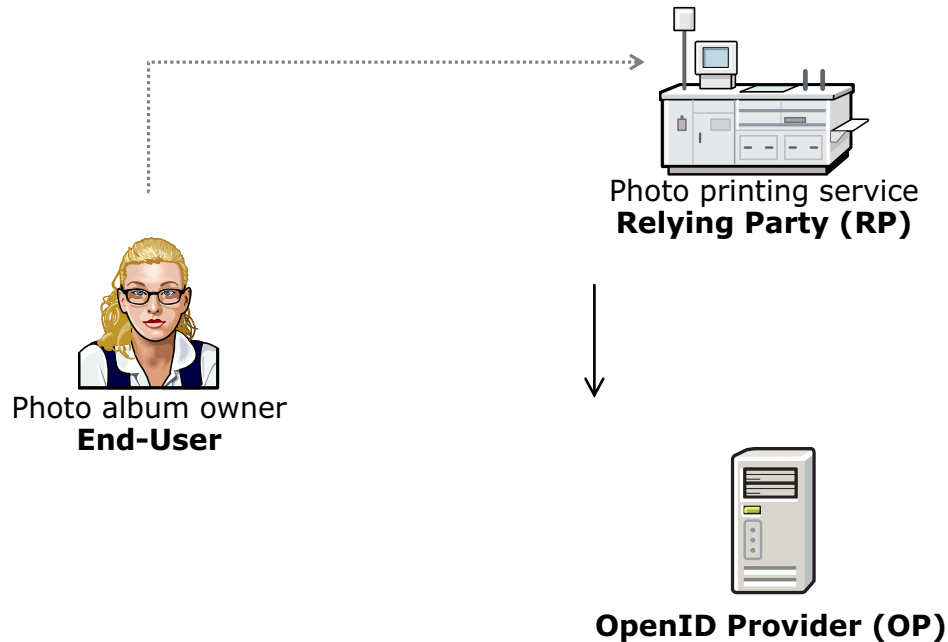
© Copyright IBM Corporation 2017

Figure 3-11. OIDC Step 1: End-user initiates request

In this scenario, Alice is the end-user. Alice wants the Print Shoppe to perform some function which requires authentication of the end-user.

OIDC Step 2: RP sends authorization request to the OP

- Step 2: The RP sends an OAuth 2.0 authorization request to the OP for end-user authentication (scope=openid)



Social Login support in DataPower

© Copyright IBM Corporation 2017

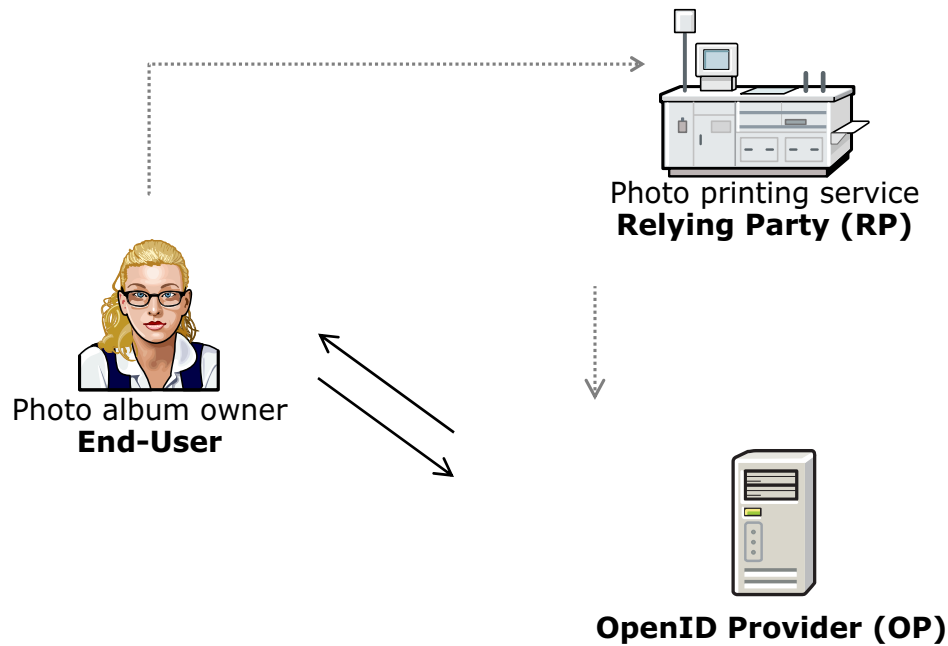
Figure 3-12. OIDC Step 2: RP sends authorization request to the OP

In the second step, the RP send an authorization request to the OP for end-user authentication. This need is indicated by the “openid” value in the scope parameter. Additional values might be in the scope parameter for standard claims retrieval.

The request also includes “response_type=code”, the client ID, and the redirect URL that is intended to receive the response.

OIDC Step 3: Authenticate end-user with the OP

- Step 3: The OP sends a login challenge to the end-user. The end-user enters her credentials.



Social Login support in DataPower

© Copyright IBM Corporation 2017

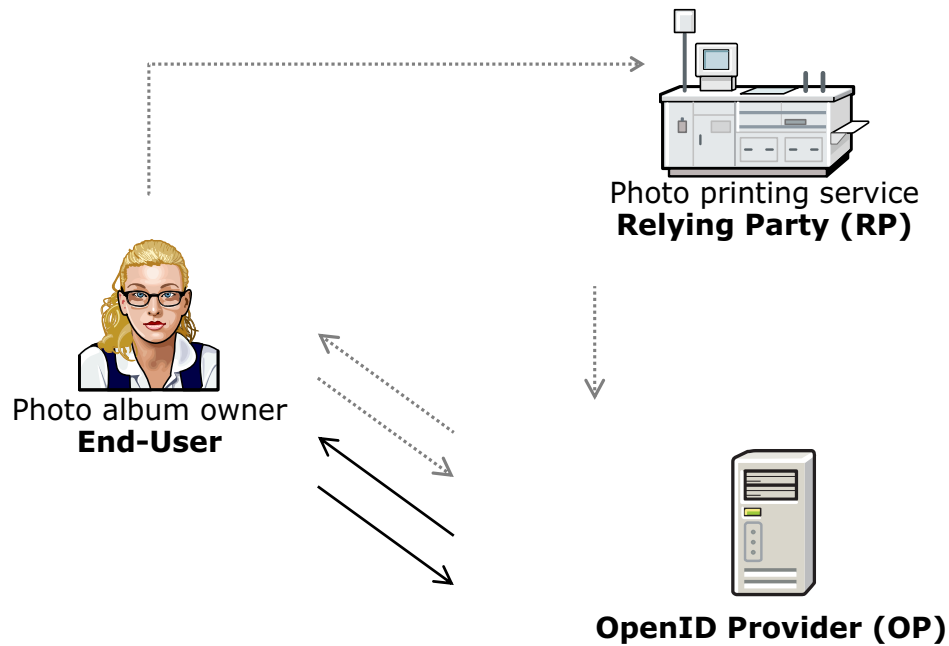
Figure 3-13. OIDC Step 3: Authenticate end-user with the OP

In the third step, the OP for Alice's user credentials to verify her identity. Alice responds with her credentials.

Notice that the ID and password interaction is between the OpenID Provider and the end-user. The Relying Party never sees that information.

OIDC Step 4: Ask end-user to give permission

- Step 4: The OP returns a web form to the end-user to give permission before releasing claims to the RP. The end-user gives consent.



Social Login support in DataPower

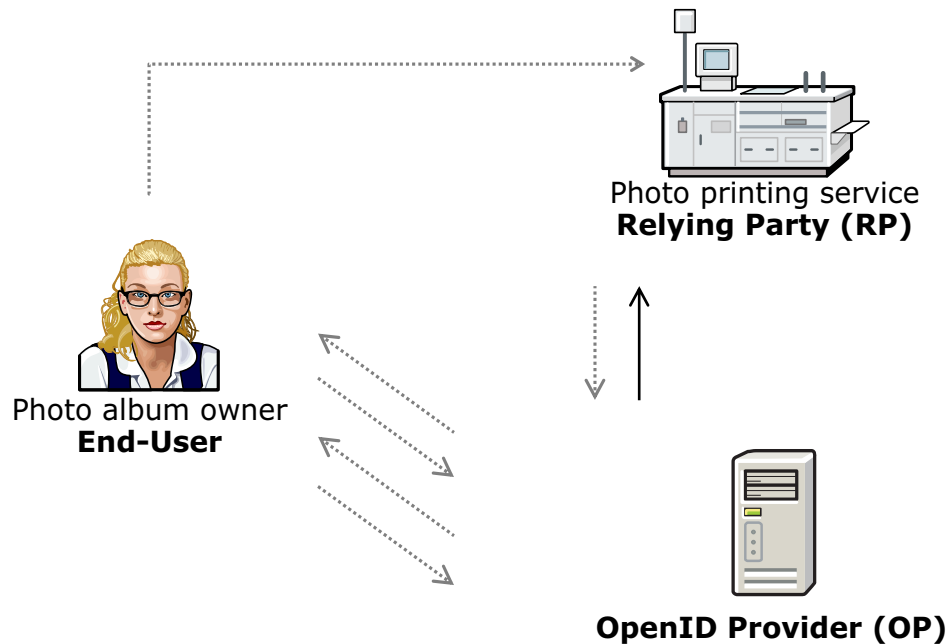
© Copyright IBM Corporation 2017

Figure 3-14. OIDC Step 4: Ask end-user to give permission

The OpenID Provider *must* obtain an authorization decision before releasing information to the Relying Party. The web form is one of the ways to obtain the end-user's permission.

OIDC Step 5: OP returns authorization response (authorization code) to the RP

- Step 5: The OP responds to the RP with the authorization code that indicates that the end-user is verified and has given permission



Social Login support in DataPower

© Copyright IBM Corporation 2017

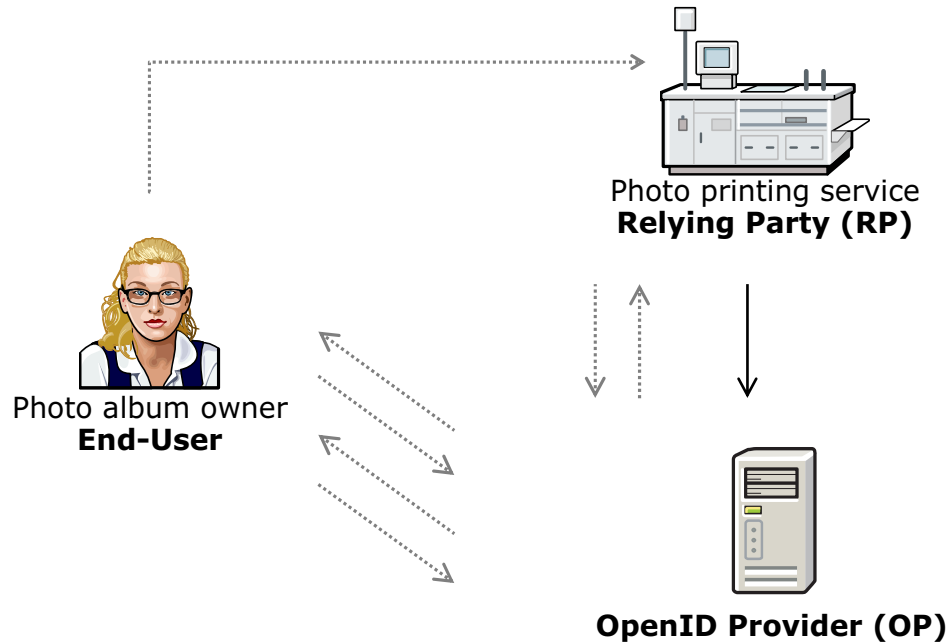
Figure 3-15. OIDC Step 5: OP returns authorization response (authorization code) to the RP

The authorization response contains the authorization code as a query parameter to the `redirect_uri` specified in the authorization request.

The OpenID Provider never transmits the end-user's user name and password to the Relying Party. Instead, the server sends an authorization grant code: a code that specifies that the end-user has authenticated to the OpenID Provider and has granted access.

OIDC Step 6: RP requests ID token, access token from OP

- Step 6: The RP sends the authorization code and client ID and password to the OP to get an ID token, access token, or refresh token



Social Login support in DataPower

© Copyright IBM Corporation 2017

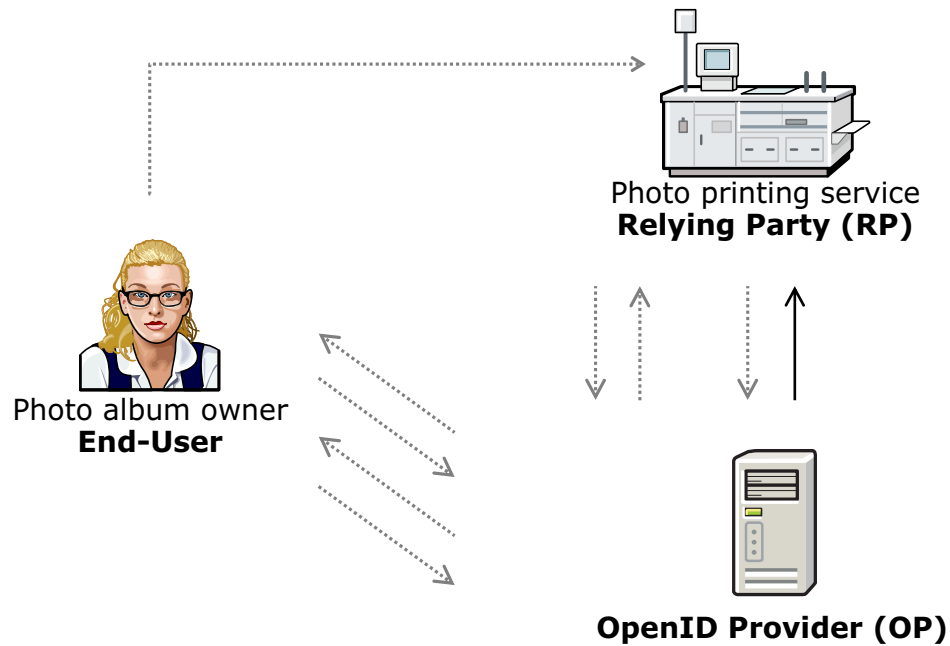
Figure 3-16. OIDC Step 6: RP requests ID token, access token from OP

The Relying Party sends its ID and password in the HTTP basic authorization header (default). The OpenID Provider authenticates the Relying Party.

The token request goes to the token endpoint of the OpenID Provider.

OIDC Step 7: OP returns access token/ID token

- Step 7: The OP validates the authorization code, and returns an access token and ID token



Social Login support in DataPower

© Copyright IBM Corporation 2017

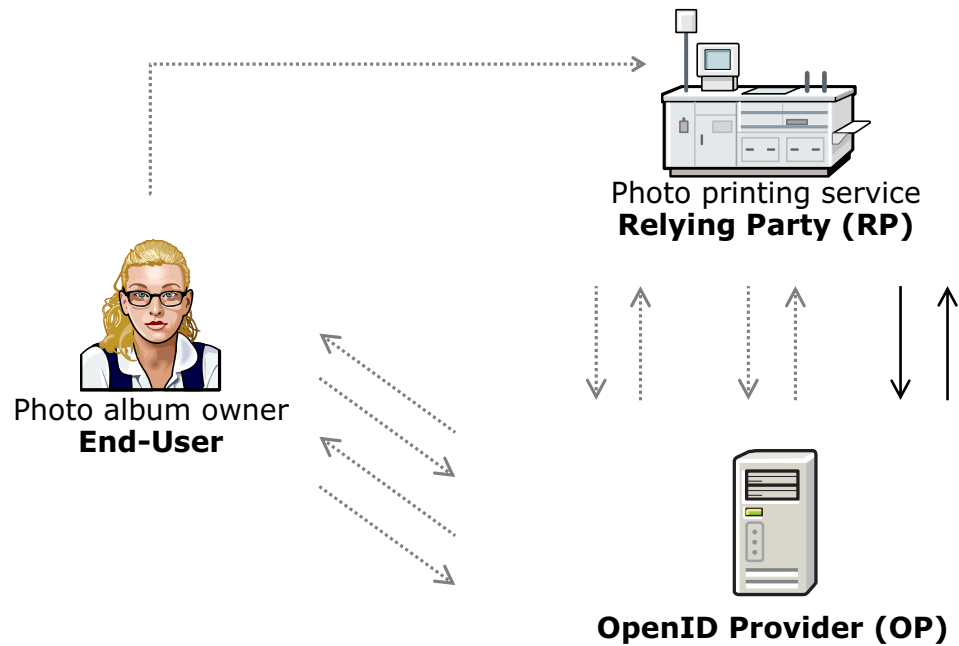
Figure 3-17. OIDC Step 7: OP returns access token/ID token

A refresh token can be returned if requested.

The ID token (JWT) and access token can then be used as needed in the Relying Party.

OIDC Optional Step: RP requests claims from the OP

- Step 8: After authentication, the RP can request more claims from the OP. The access token is required. A JSON object of claims is returned.



Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-18. OIDC Optional Step: RP requests claims from the OP

The returned claims are formatted as a JSON object.

OIDC support in DataPower V7.5.1

- You can use the social login support in the following ways:
 - Use it alone in the AAA policy to redirect all incoming requests to a social login provider for authentication support (DataPower as an OIDC client)
 - Use it within the OAuth protocol in the AAA policy to add authentication support
- The OIDC or social login support is enabled by
 - Options in the AAA Policy object
 - Social Login Policy object
 - Contains OIDC client information that DataPower uses to interact with the OP
 - Web Token Service (can be an OP)
 - JWT Generator object
 - Create, sign, encrypt a JWT
 - JWT Validator object
 - Validate a received JWT

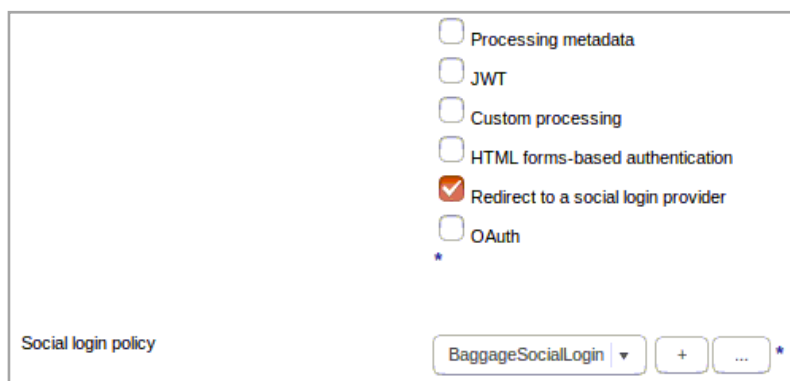
Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-19. OIDC support in DataPower V7.5.1

Options in the AAA Policy object: Identity extraction

- Specify use of a social login provider
 - Also enables the **Social login policy** selection
- At end of extraction, an ID token (JWT) is returned
 - The “sub” (subject) in the JWT becomes the AAA **username**
- Notice the **JWT** option to get the user identity from a JWT
 - Causes a **JWT Validator** selection to display



The screenshot shows a configuration window for the 'Social login policy'. On the right side, there is a list of options with checkboxes: 'Processing metadata', 'JWT', 'Custom processing', 'HTML forms-based authentication', 'Redirect to a social login provider' (which is checked), and 'OAuth'. A small blue asterisk is located below the 'OAuth' option. At the bottom left of the window, the text 'Social login policy' is displayed. At the bottom right, there is a dropdown menu showing 'BaggageSocialLogin', followed by a '+' button, an '...' button, and a '*' button.

Social Login support in DataPower

© Copyright IBM Corporation 2017

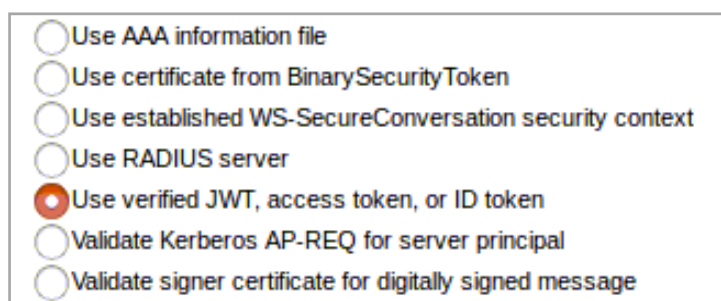
Figure 3-20. Options in the AAA Policy object: Identity extraction

The Social Login Policy object is covered later in this unit.

The JWT Validator object is covered later in this unit.

Options in the AAA Policy object: Authenticate user

- Can specify **Use verified JWT, access token, or ID token**
 - JWT validator is specified as part of the Social login policy
- You can still use other authentication methods such as LDAP
 - An option in the JWT validator specifies which claim in the JWT is to be treated as the username
 - For example, you can have the username be the returned email address, and use the email address to authenticate to an LDAP server



A screenshot of a configuration window titled 'Options in the AAA Policy object: Authenticate user'. It contains a list of seven radio button options. The third option, 'Use verified JWT, access token, or ID token', is selected, indicated by a red dot in the center of the radio button. The other options are unselected, with empty radio buttons.

- ☐ Use AAA information file
- ☐ Use certificate from BinarySecurityToken
- ☐ Use established WS-SecureConversation security context
- ☐ Use RADIUS server
- ☒ Use verified JWT, access token, or ID token
- ☐ Validate Kerberos AP-REQ for server principal
- ☐ Validate signer certificate for digitally signed message

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-21. Options in the AAA Policy object: Authenticate user

Options in the AAA Policy object: Postprocessing

- The AAA postprocessing phase can specify generation of more tokens such as SAML tokens and WS-Security tokens
- Another option is to generate a JWT
 - If enabled, a field displays to designate a JWT Generator object

Generate a JSON Web Token	<input checked="" type="radio"/> on <input type="radio"/> off
JSON Web Token setting	CreateBaggageJWT ▼ +

Social Login support in DataPower

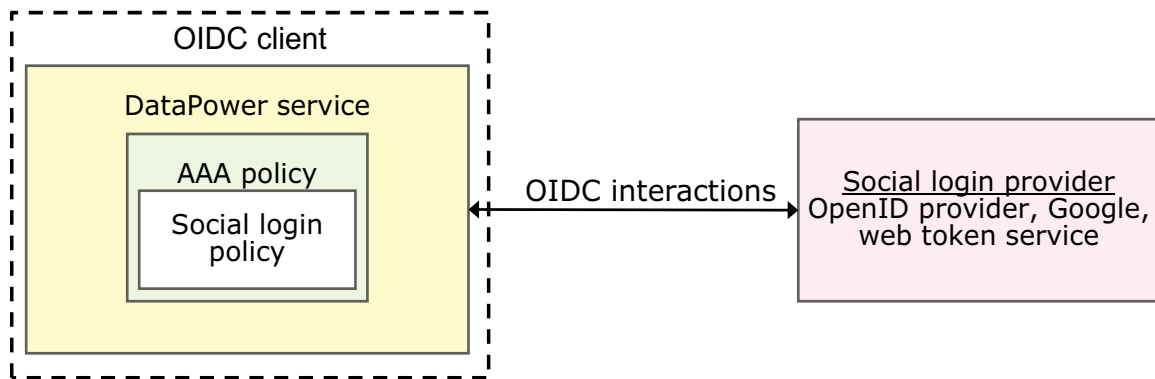
© Copyright IBM Corporation 2017

Figure 3-22. Options in the AAA Policy object: Postprocessing

The JWT Generator object is covered later in this unit.

Social login policy object

- The Social login policy object defines the variables and interaction between the service (which contains the AAA policy) and the social login provider
 - Provides the details that are needed to act as the OIDC client



Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-23. Social login policy object

Social login policy configuration: Top

Social Login Policy: BaggageSocialLogin [up]

Apply Cancel Undo

General

Administrative state ☒ enabled ☐ disabled

Comments

Client Settings

Client ID 1 BaggageOIDCClient

Client secret 2 password

Client grant type 3 Authorization Code

Scope 4 openid RequestBagsForFlight

Client redirection URI 5 URL-in/social-login-callback

Client Optional Query Parameters

SSL client profile ssl-client + ...

1. The **client ID** and **client secret** (password) that are presented to the social login provider

2. **Authorization Code** is the only supported grant type for social login

3. The **scopes** that are requested from the social login provider
- **openid** indicates OIDC

4. The **client redirection URI** specifies the endpoint on DataPower that the social login provider uses in the callback

5. The **client optional query parameters** specifies more query parameters to include in the request that is sent to the social login provider

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-24. Social login policy configuration: Top

The client ID and client secret that are specified here must match the values that are specified in the social login provider.

- If the provider is a web token service, the OAuth client profile that it references contains the client ID and password.
- If the provider is Google, Google generates a client ID and password. The social login policy values must match.
- For other providers, it depends on their implementation.

For the **Scope** field, multiple values can be specified. They must be separated by blanks.

The client redirection URL is sent to the provider so that the provider has the correct callback location. The value *must* end in “/social-login-callback”.

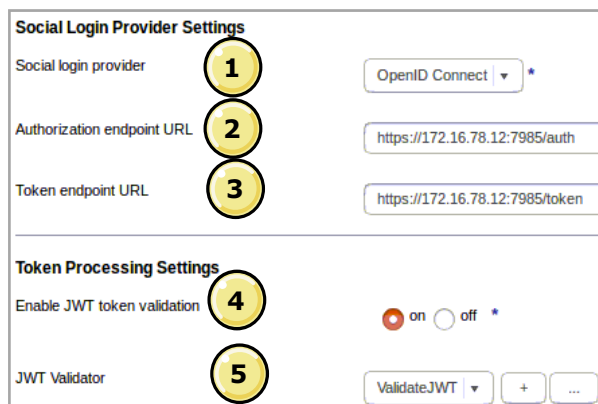
The URL can be specified in the following ways:

- A hardcoded string.
- A “URL-in/social-login-callback”. The “URL-in” is the standard service variable.
- A DataPower context variable.

The optional query parameters are defined in Section 3.1.2.1 of the OpenID Connect specification. Some examples are **nonce** (a random text string to mitigate replay attacks) and **display** (how the authentication/consent screen is displayed to the end user).

The SSL client profile is used to establish the SSL connection with the provider.

Social login policy configuration: Bottom



Social Login Provider Settings

Social login provider **1** OpenID Connect *

Authorization endpoint URL **2** https://172.16.78.12:7985/auth

Token endpoint URL **3** https://172.16.78.12:7985/token

Token Processing Settings

Enable JWT token validation **4** ☒ on ☐ off *

JWT Validator **5** Validate.JWT + ...

1. The **Social login provider** indicates that the provider is a generic OpenID Connect provider or Google

2. **Authorization endpoint URL** of the provider

3. The **Token endpoint URL** of the provider

4. **Enable JWT token validation** or not

5. The name of the **JWT Validator** object that specifies the parameters for JWT validation

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-25. Social login policy configuration: Bottom

If the social login provider is specified as Google, it means that DataPower processes OIDC transactions as specified by Google in <https://developers.google.com/identity/protocols/OpenIDConnect>.

If the provider is Google, get the authorization and token endpoint URL values from the same Google URL.

The suggestion is to validate a JWT.

The JWT Validator object is covered later in this unit.

JWT validator object: Top of Main tab

JWT Validator: Validate.JWT [up]

Apply Cancel Undo

General

Administrative state ☒ enabled ☐ disabled

Comments

Issuer **1** BaggageWTS

Audience **2** BaggageOIDClient

Validation method **3** ☒ Decrypt ☒ Verify ☒ Custom processing

Custom validation method processing **4** local:/// (none) Upload... Fetch...

1. The **Issuer ("iss") is the creator of the JWT**

2. The **Audience ("aud") for the JWT**

- Must match or the validation fails

3. The **Validation method options enable other options on the page**

4. Custom validation method processing identifies a stylesheet/GatewayScript to perform further custom processing

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-26. JWT validator object: Top of Main tab

The **Issuer** is a PCRE string that verifies the issuer claim in the JWT.

The **Audience** is a PCRE string that verifies the audience claim in the JWT.

The **Validation method** specifies what validation to perform on the JWT:

- **Decrypt** indicates that the JWT must be decrypted successfully
- **Verify** indicates that the JWT must be validated successfully
- **Custom processing** indicates that a stylesheet or GatewayScript executes after the decryption and validation succeeds

JWT validator object: Bottom of Main tab

Decrypt the token

Decrypt method **1** PKIX

Decrypt key **2** (none) + ...

Verify the token

Verify method **3** PKIX

Verify certificate **4** AliceCert + ...

Signature validation credentials **5** ☒ on ☐ off *

Validation credentials **6** (none) + ...

- PKIX
 - Use crypto key/certificate
- Remotely retrieve JWK
 - Use URL to retrieve JWK
- Shared secret key
 - Use shared secret key
- Custom
 - XSL/GatewayScript for postprocessing

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-27. JWT validator object: Bottom of Main tab

1. Decrypt method (if enabled)

- PKIX
- Remotely retrieve JWK
- Shared secret key
- custom

2. Crypto key for PKIX decryption

3. Verify method (same as Decrypt method)

4. Verify certificate is the crypto certificate for validation

5. Signature validation credentials enables use of a validations credential

6. Validation credentials identifies the crypto validation credentials to use, if specified

Custom processing runs only after a successful decryption or validation.

JWT validator object: Advanced tab

JWT Validator: ValidateJWT [up]

Apply Cancel Undo

Validate claims

Name	Value	Type
email	^[a-zA-Z0-9+&*-]+(?:\\.[a-zA-Z0-9+&*-]+)*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,7}\$	String

Claim used as username email

1. Validate claims specifies any other claims that must be present in the JWT to be considered as a valid JWT

- **Name** is a text string of the claim label
- **Value** is the test string for the claim (PCRE)
- **Type** is String, Number, or Boolean

2. email is an example of a test for a claim to contain a valid email address

3. Claim used as username identifies the claim that is used as the extracted identity (username) for the AAA policy

- **sub** (subject) is the default

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-28. JWT validator object: Advanced tab

PCRE is “Perl Compatible Regular Expressions”.

The sample PCRE for the email claim

(^[a-zA-Z0-9+&*-]+(?:\\.[a-zA-Z0-9+&*-]+)*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,7}\$) tests for a validly formatted email address.

JWT generator object: Top of Main tab

- A service might need to generate a JWT:
 - Postprocessing phase of AAA policy
 - Generate a JWT as output from an OAuth client profile
 - Role of OAuth client includes “OpenID Connect”

JWT Generator: CreateBaggageJWT [up]

Apply Cancel Undo

Administrative state ☒ enabled ☐

Comments

Issuer **1** BaggageWTS

Validity period **2** 3600

JWT generation method **3** ☒ Sign ☒ Encrypt

1. Issuer indicates the principal that issues the JWT, the “iss” claim

- Default is “idg”

2. Validity period (in seconds) of the JWT

3. JWT generation method indicates what security to apply to the JWT

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-29. JWT generator object: Top of Main tab

The validity period is used to compute the expiration time (“exp”) claim.

JWT generator object: Bottom of Main tab

1. Signing algorithm indicates the HMAC, RSA, or ECDSA algorithm to use

2. Signing key is a crypto key that is used for RSA or ECDSA

- **Shared secret key** if HMAC

3. Encryption algorithm indicates the AES_HMAC algorithm to use

4. Key management algorithm indicates the RSA or AES Key Wrap algorithm to use

5. Encryption certificate is the crypto certificate that is used to encrypt the key

- **Signing key** (crypto shared secret key) if AES Key Wrap

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-30. JWT generator object: Bottom of Main tab

The signing activity creates a JWS.

The encrypting activity creates the JWE.

The signing algorithm choices are HMAC using SHA-256, HMAC using SHA-384, HMAC using SHA-512, RSASSA-PKCS-v1_5 using SHA-256, RSASSA-PKCS-v1_5 using SHA 384, RSASSA-PKCS-v1_5 using SHA-512, ECDSA using P-256 and SHA-256, ECDSA using P-384 and SHA-384, and ECDSA using P-521 and SHA-512.

The encryption algorithms choices are AES_128_CBC_HMAC_SHA_256 authenticated encryption algorithm, AES_192_CBC_HMAC_SHA_384 authenticated encryption algorithm, and AES_256_CBC_HMAC_SHA_512 authenticated encryption algorithm.

The key management algorithm choices are RSAES-PKCS1-V1_5, RSAES OAEP using default parameters, RSAES OAEP using SHA-256 and MGF1 with SHA-256, AES Key Wrap with default initial value using 128 bit key, AES Key Wrap with default initial value using 192 bit key, AES Key Wrap with default initial value using 256 bit key, and direct use of a shared symmetric key as the CEK.

JWT generator object: Advanced tab

1. Additional claims (besides "sub") to include in the JWT:

- "aud"
- "nbf"
- "iat"
- "jti"
- "nonce"
- **Custom** enables **Custom claims** option

2. Audience claim identifies the recipient(s) of the JWT

3. Custom claims specifies an XSLT/GatewayScript to add more claims to the JWT

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-31. JWT generator object: Advanced tab

The **subject** ("sub") claim is included in the JWT by default.

The "jti" claim is calculated from "session.parameter.uuid".

The custom claims XSLT or GatewayScript can be used to override the default "sub" claim.

Google as the social login provider

- Google OIDC behavior is documented:
 - <https://developers.google.com/identity/protocols/OpenIDConnect>
- The current details of the Google's configuration for OIDC are documented in a "discovery" document, which is discussed at:
 - <https://developers.google.com/identity/protocols/OpenIDConnect#discovery>
- The current discovery document is a JSON document at:
 - <https://accounts.google.com/.well-known/openid-configuration>
 - Current partial capture:

```
{
  "issuer": "https://accounts.google.com",
  "authorization_endpoint": "https://accounts.google.com/o/oauth2/v2/auth",
  "token_endpoint": "https://www.googleapis.com/oauth2/v4/token",
  "userinfo_endpoint": "https://www.googleapis.com/oauth2/v3/userinfo",
  "revocation_endpoint": "https://accounts.google.com/o/oauth2/revoke",
  "jwks_uri": "https://www.googleapis.com/oauth2/v3/certs",
  "response_types_supported": [
    "code",
    "token",
    "id_token",
    "code_token"
  ]
}
```

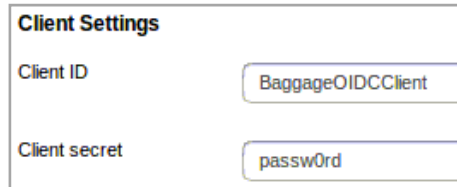
- Always verify the current endpoints

The discovery document is part of the OpenID Connect specification.

The screen captures reflect the current values of the endpoints at the time of creation.

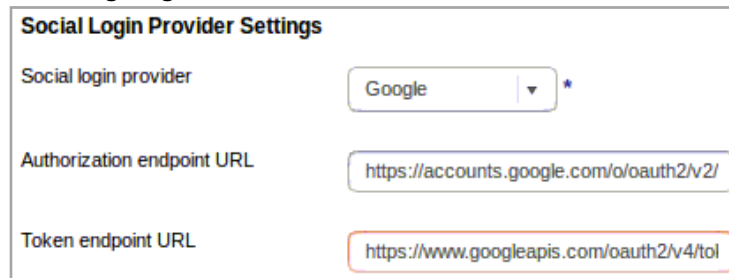
Google as the social login provider: Social Login Policy

- Google OIDC behavior is documented:
 - <https://developers.google.com/identity/protocols/OpenIDConnect>
- Client ID and secret values come from Google API settings



A screenshot of a 'Client Settings' form. It contains two rows: 'Client ID' with a text input field containing 'BaggageOIDCClient', and 'Client secret' with a text input field containing 'passw0rd'.

- Google has published authorization and token endpoint URLs
 - <https://accounts.google.com/o/oauth2/auth>
 - <https://accounts.google.com/o/oauth2/token>



A screenshot of a 'Social Login Provider Settings' form. It contains three rows: 'Social login provider' with a dropdown menu showing 'Google' and a blue asterisk icon; 'Authorization endpoint URL' with a text input field containing 'https://accounts.google.com/o/oauth2/v2/'; and 'Token endpoint URL' with a text input field containing 'https://www.googleapis.com/oauth2/v4/tol'.

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-33. Google as the social login provider: Social Login Policy

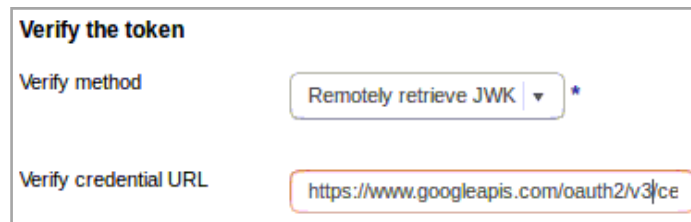
Where is the social login information in Google?

- Log in to Google's developer console
 - <https://console.developers.google.com>
- Select **Credentials** and click your **OAuth 2.0 Client ID**
 - Shows the Google generated client ID and client secret

Client ID for Web application	
Client ID	774187843246-ggmr27j3rgervbk6jj0mbadf0e2vmd12.apps.googleusercontent.com
Client secret	wjI837NAqf96edbfItX4998
Creation date	Jan 25, 2017, 12:37:51 PM
Name	
GoogleSocialLoginClient	

Getting Google's public certificate

- JWT Validator object
 - To verify a JWT from Google, you need the Google certificate URL to retrieve the JWK
 - <https://www.googleapis.com/oauth2/v3/certs>



Verify the token

Verify method: Remotely retrieve JWK *

Verify credential URL: <https://www.googleapis.com/oauth2/v3/certs>

Caching the Google's public certificate

- Google refreshes its certificate/JWK frequently
 - Use a DataPower document cache to reduce the number of network calls
- Specify the **URL Match Expression** to match the **Verify credential URL** in the JWT Validator object
 - <https://www.googleapis.com/oauth2/v3/certs>

Configure XML Manager

[Main](#) | [XML Parser](#) | [Document Cache](#) | [Extension Functions](#) | **[Document Cache Policy](#)** | [Schema Validation Rules](#) | [Scheduled Processing Policy Rules](#)

XML Manager: default [up]

[Export](#) | [View Log](#) | [View Status](#) | [Help](#)
[Flush Stylesheet Cache](#) | [Flush Document Cache](#) | [Flush LDAP Cache](#)

Document Cache Policy

URL Match Expression	Policy Type	TTL	Priority	Cache Grid	Cache Back-end Responses	HTTP Cache Validation	Return Expired Document	RESTful Invalidation	Cache Response to POST and PUT Requests	
https://www.googleapis.com/oauth2/v3/certs	Protocol-Based	900	128		off	off	off	off	off	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-36. Caching the Google's public certificate

Recovering the initial URI: The need

- DataPower AAA policy uses two passes to manage the OIDC interaction
- During second pass, service/URI and service/URL-in contain the callback from the social login provider
 - Original URL from user/browser is no longer in those service variables

var://service/URI	string	/RequestBagInfo/social-login-callback?code=AALsvyI0vOIF0eyKub9_ahhteOLVvJqAvalnrSfvzoSKP-JuUzWjknDrTvJqN6sZOJAWNstate=99580514FEDE01758465C580FFF29608D4
var://service/URL-in	string	'https://172.16.78.12:7994/RequestBagInfo/social-login-callback?code=AALsvyI0vOIF0eyKub9_ahhteOLVvJqAvalnrSfvzoSKP-JuUzWjknDrTvJqN6sZOJAWNstate=99580514FEDE01758465C580FFF29608D4'

- If you need the original URI information (path and request parameters), you need to explicitly “recall” it from a AAA context variable
 - var://context/AAA/social-login-url-in

```
var://context
/AAA/social-login- string 'https://172.16.78.12:7994/RequestBagInfo?bagId=1289'
url-in
```

Recovering the initial URI: Example

- Use a GatewayScript after the OIDC-focused AAA action



```
// Retrieve original URL from AAA context
var ctx = session.name('AAA');
var originalURL = ctx.getVar('social-login-url-in');

if (typeof originalURL != 'undefined') {
  console.debug('Original URL: ' + originalURL);
  . . .
}
```

- `if (typeof originalURL != 'undefined')` is used to bypass processing in first pass through AAA policy

Combining social login support (OIDC) with OAuth

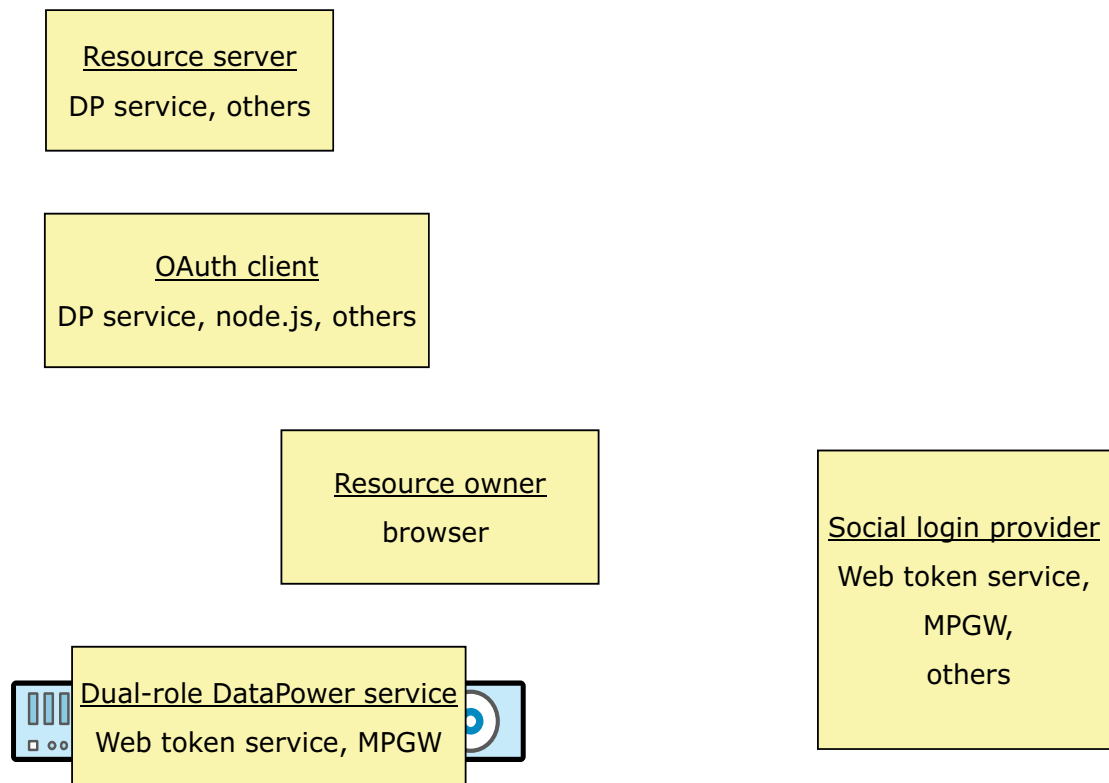
- Recall that OIDC is an extension of the OAuth 2.0 protocol
 - DataPower supports OIDC and OAuth combination
- A single service in DataPower plays two roles in this combination
 - OAuth 2.0 authorization server
 - OIDC client
- An inner and an outer protocol flow
 - “Outside” flow is the OAuth flow
 - Within the OAuth flow, the OIDC flow occurs
- Being a participant in the two flows is why the single DataPower service plays two roles
- This service has the Extract Identity phase of the AAA policy that specifies **OAuth** and **redirect to social login provider**

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-39. Combining social login support (OIDC) with OAuth

Roles in the OAuth-OIDC combination

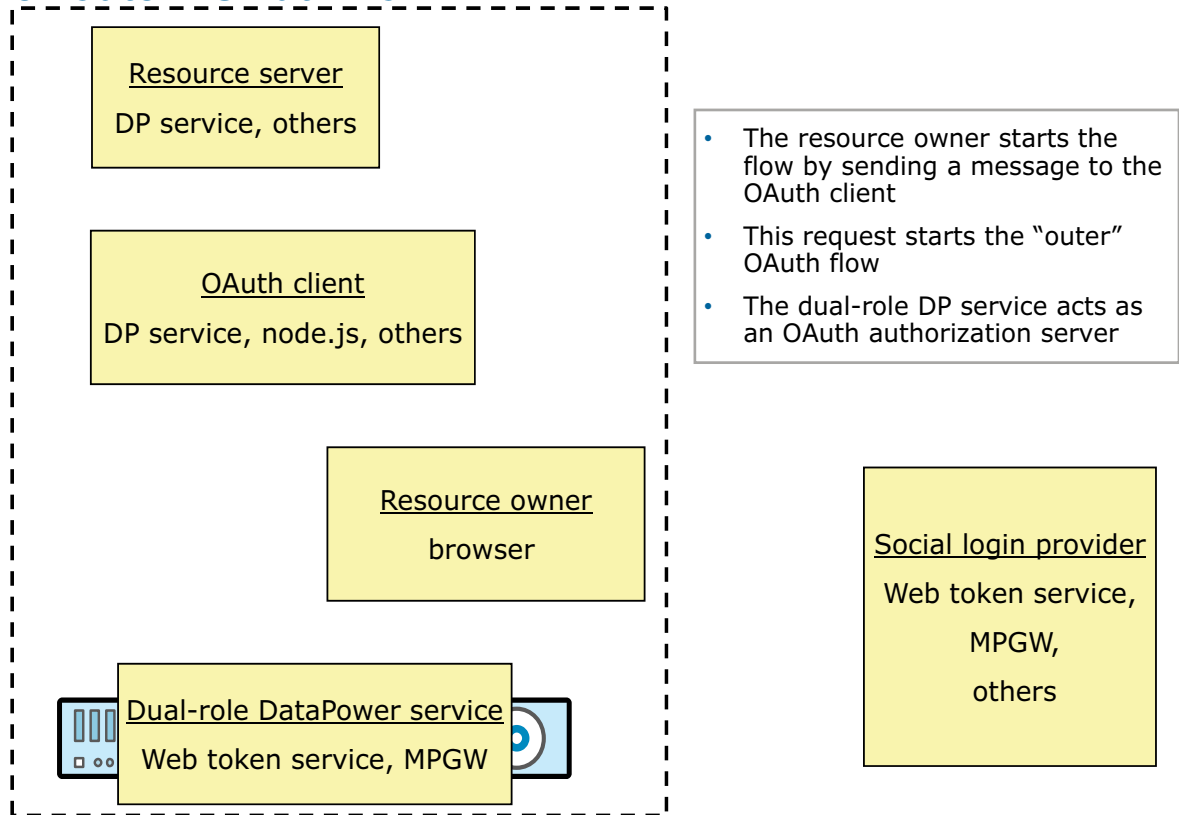


Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-40. Roles in the OAuth-OIDC combination

The “outer” OAuth flow

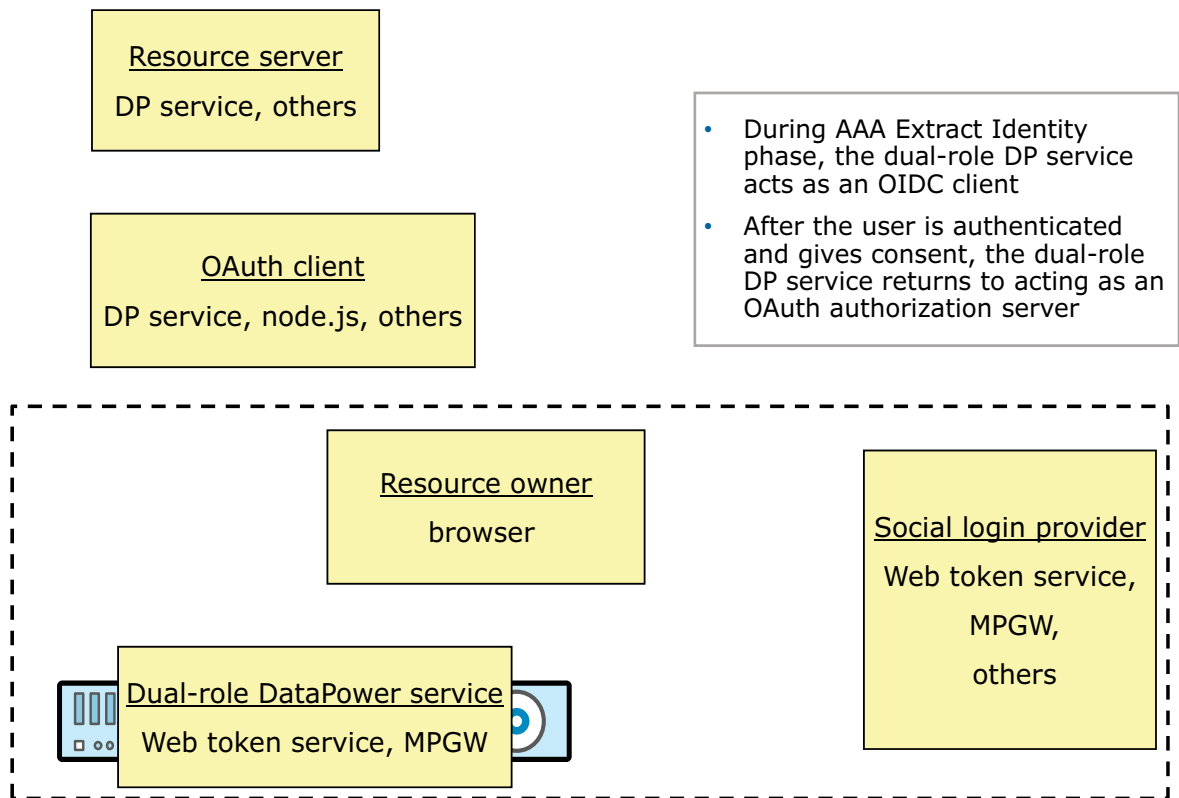


Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-41. The “outer” OAuth flow

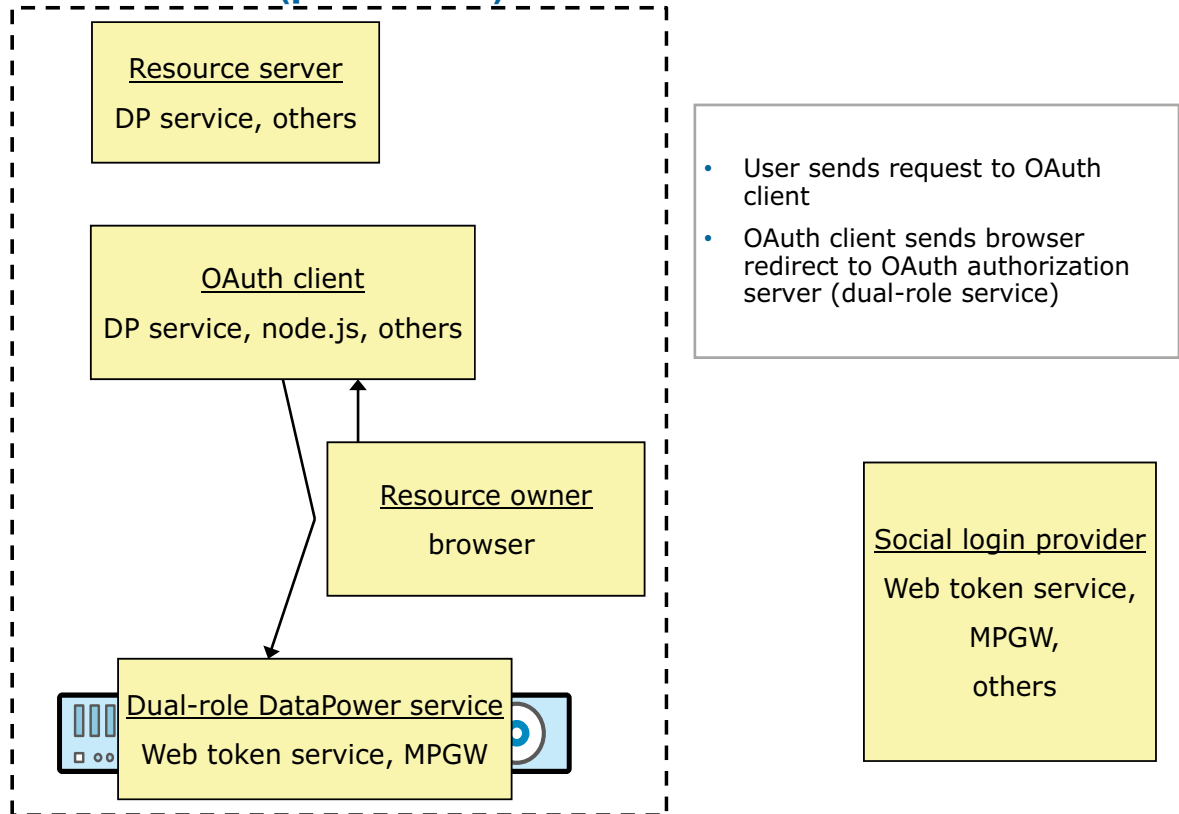
The “inner” OIDC flow



Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-42. The “inner” OIDC flow

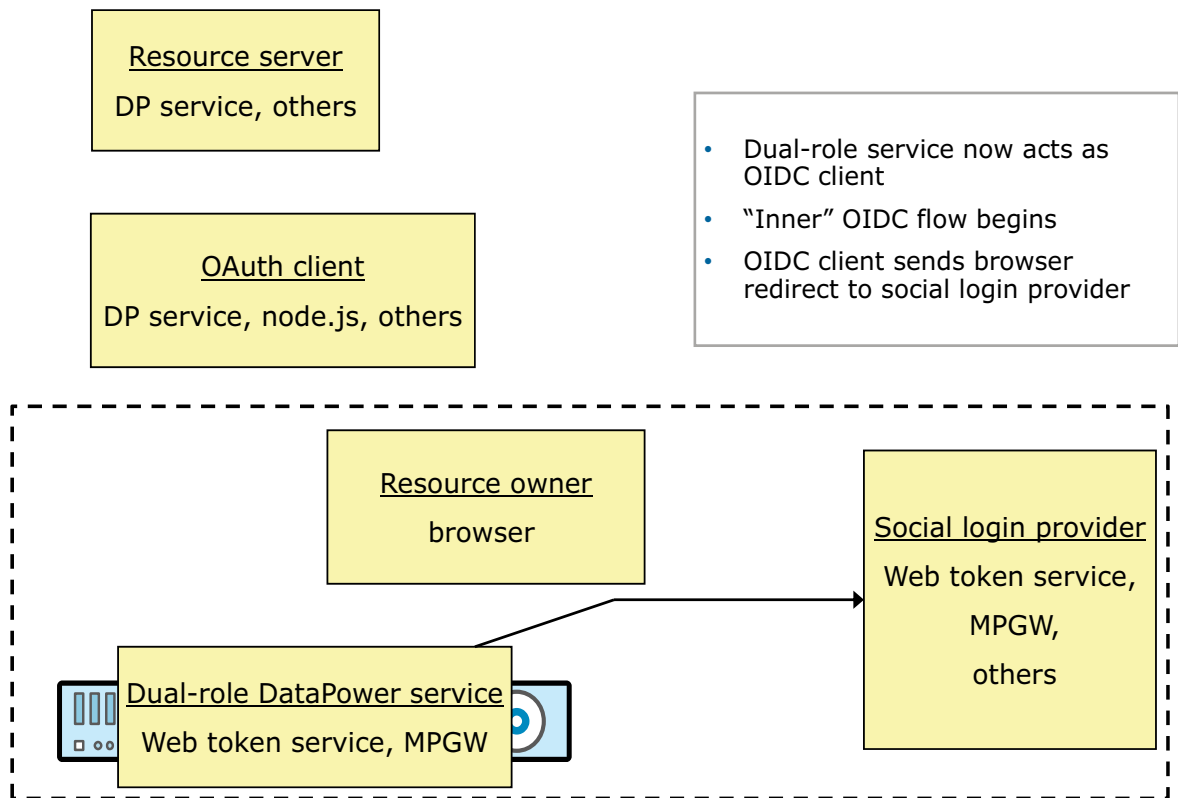
Combined flow (part 1 of 7)

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-43. Combined flow (part 1 of 7)

Combined flow (part 2 of 7)

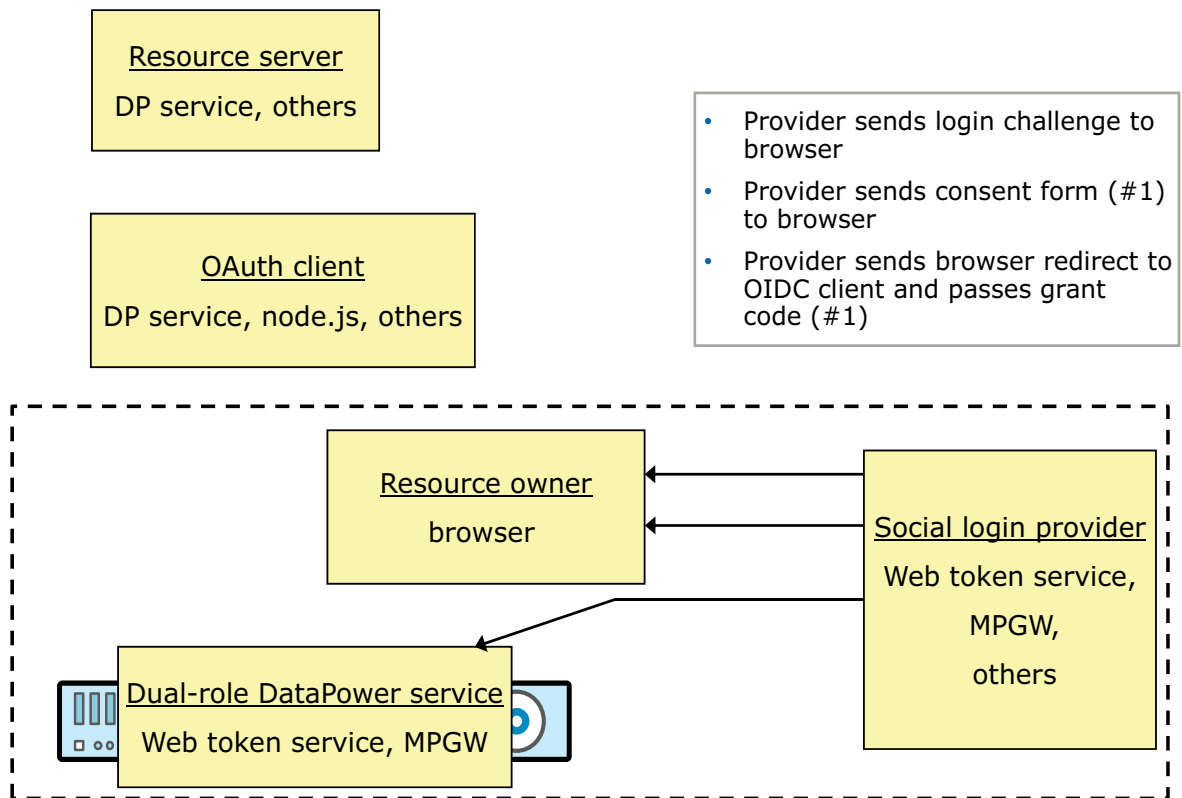


Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-44. Combined flow (part 2 of 7)

Combined flow (part 3 of 7)

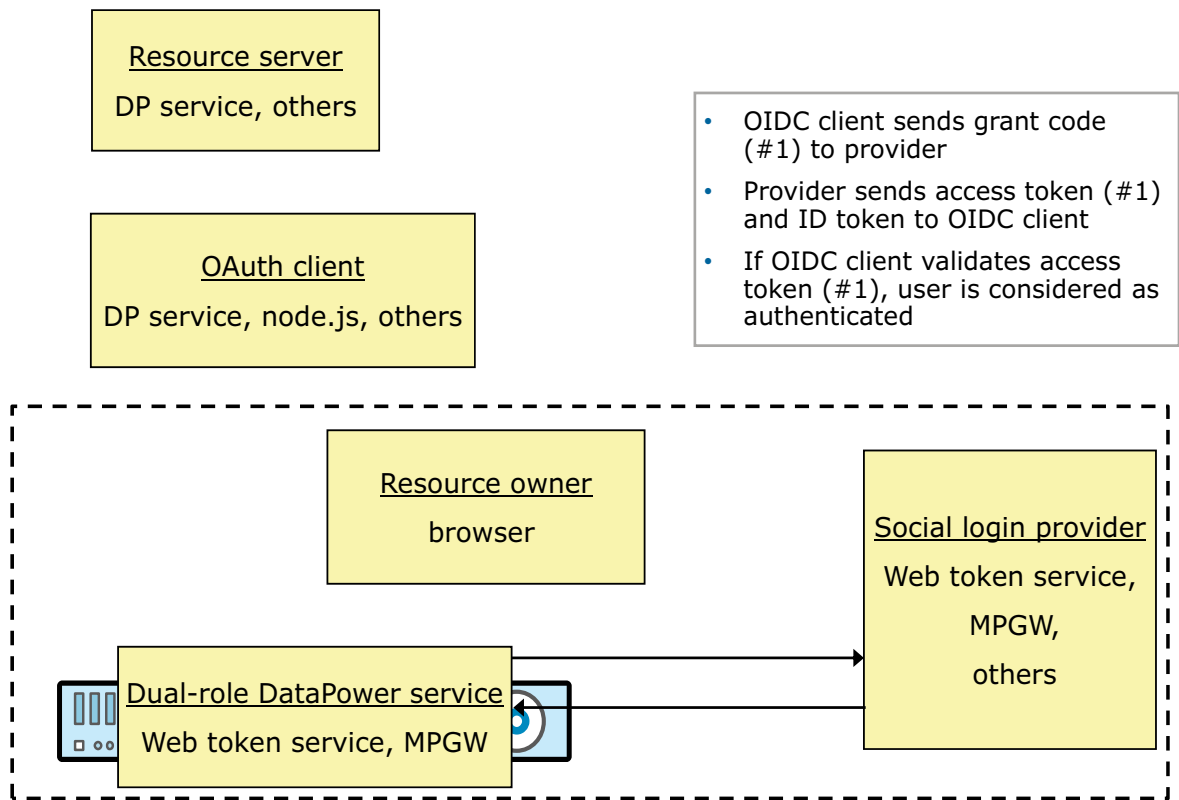


Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-45. Combined flow (part 3 of 7)

Combined flow (part 4 of 7)



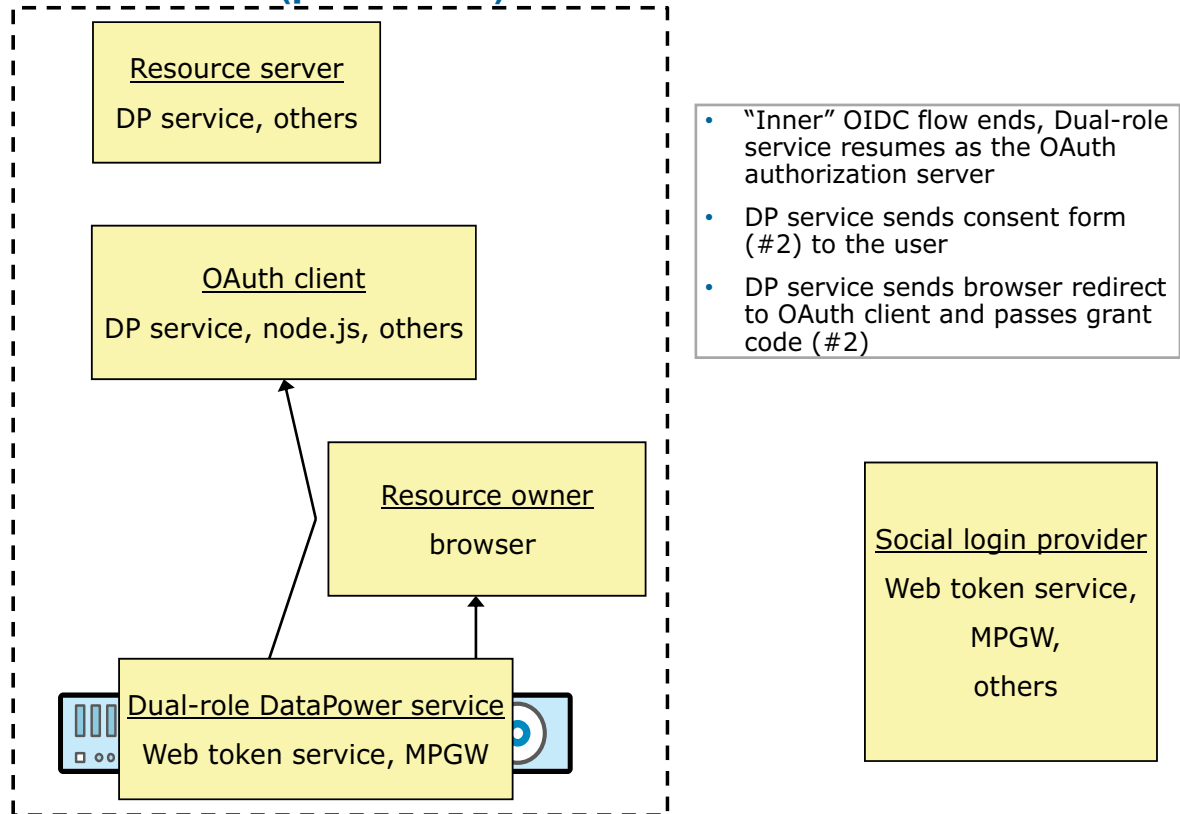
Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-46. Combined flow (part 4 of 7)

When the OIDC client sends the grant code to the provider, it also sends a client ID and password. This information authenticates the client to the provider.

Combined flow (part 5 of 7)

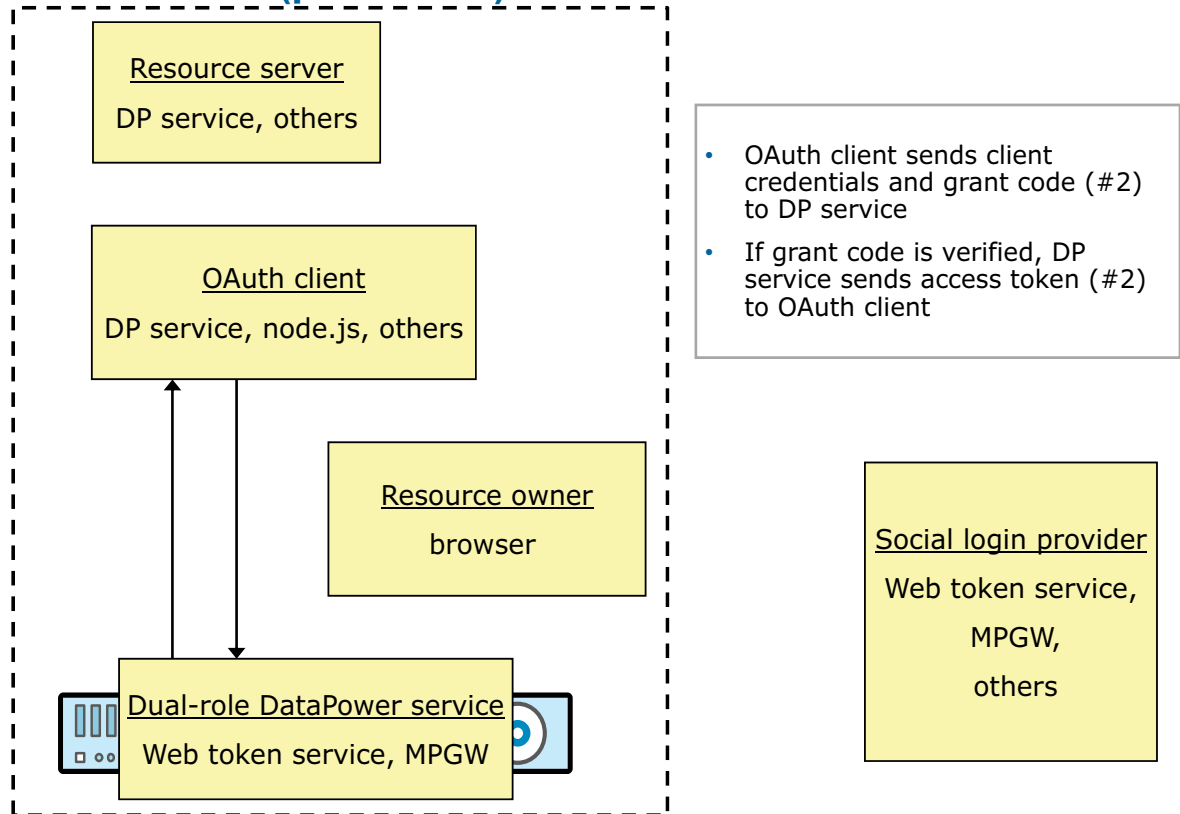


Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-47. Combined flow (part 5 of 7)

This authorization grant code is different than the code that was sent from the provider and the DataPower service.

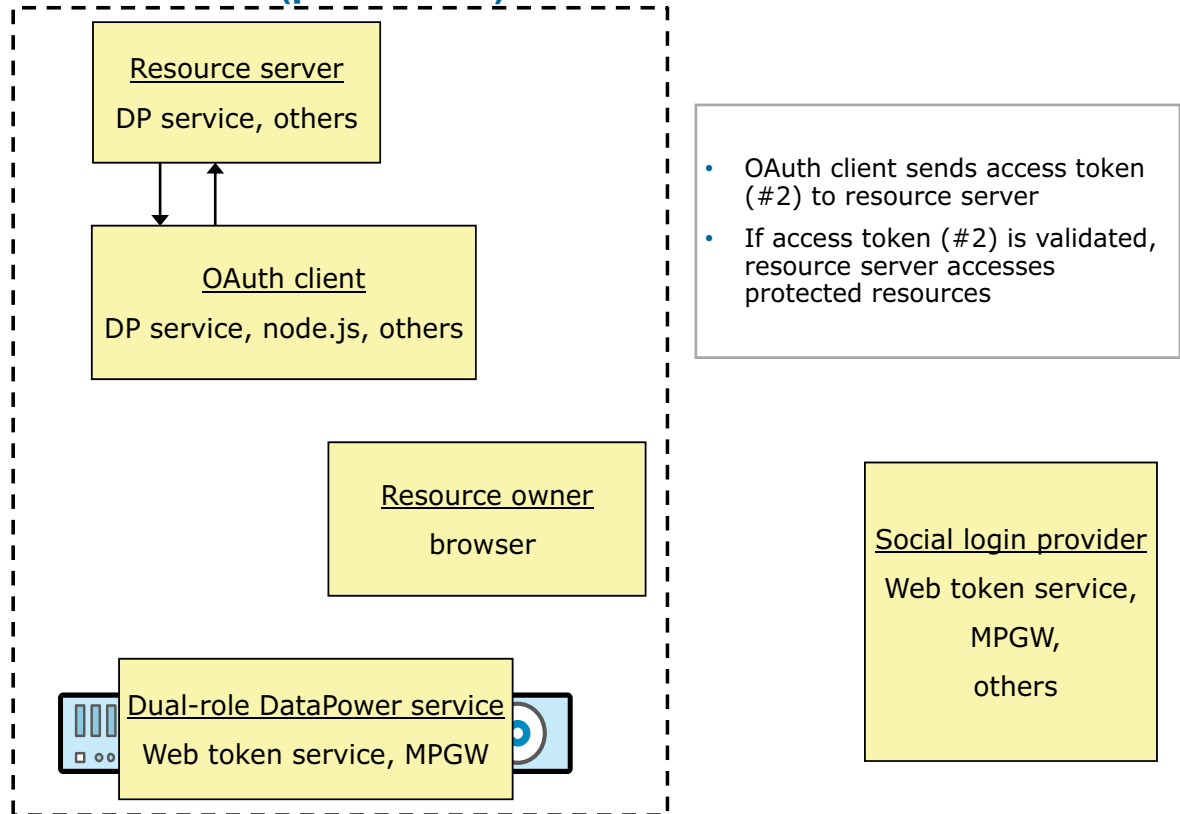
Combined flow (part 6 of 7)

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-48. Combined flow (part 6 of 7)

This access token is different than the access token sent from the provider to the DataPower service. Access token (#2) represents the OAuth client access to the resource server.

Combined flow (part 7 of 7)

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-49. Combined flow (part 7 of 7)

Unit summary

- Define Social Login
- Describe how to configure Social Login in DataPower
- Configure a Social Login Policy object
- Configure a JWT Generator and a JWT Validator object
- Describe OpenID Connect
- Configure an OpenID Connect client in DataPower

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-50. Unit summary

Review questions



1. True or False: DataPower V7.5.1 supports all forms of social login.
2. True or False: All requested claims in the ID token must be requested during the initial authentication and authorization phase.
3. Which object specifies the client ID and password that is sent from the OIDC client to the provider:
 - A. OAuth client profile
 - B. Web token service
 - C. Social login policy
 - D. JWT generator
 - E. OAuth Client Profile and OAuth Client Group
4. True or False: The OIDC client never sees the login credentials of the user.

Social Login support in DataPower

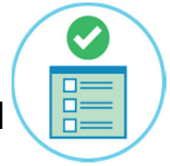
© Copyright IBM Corporation 2017

Figure 3-51. Review questions

Write your answers here:

- 1.
- 2.
- 3.
- 4.

Review answers (1 of 2)



1. True or False: DataPower V7.5.1 supports all forms of social login.

The answer is False. DataPower V7.5.1 handles social login only for providers that implement OpenID Connect. Some providers, such as Facebook, do not implement OIDC.

2. True or False: All requested claims in the ID token must be requested during the initial authentication and authorization phase.

The answer is False. Claims can be independently requested from the Userinfo endpoint of the social login provider.

Review answers (2 of 2)



3. Which object specifies the client ID and password that is sent from the OIDC client to the provider:

- A. OAuth client profile
- B. Web token service
- C. Social login policy
- D. JWT generator
- E. OAuth Client Profile and OAuth Client Group

The answer is C. The social login policy object specifies the client ID and password that are sent to the provider to authenticate itself. If the provider is Google, you obtain the client ID and password from your Google API Manager page. If the provider is a web token service or MPGW, the client ID and password is referenced on the provider side from the OAuth client profile.

4. True or False: The OIDC client never sees the login credentials of the user.

The answer is True. The OIDC login credentials are shared between the user and the provider only. This isolation is similar to that of OAuth.

Exercise: Implementing an OIDC client

Social Login support in DataPower

© Copyright IBM Corporation 2017

Figure 3-54. Exercise: Implementing an OIDC client

Exercise objectives

- Configure an OIDC client



Social Login support in DataPower

Figure 3-55. Exercise objectives

© Copyright IBM Corporation 2017

Unit 4. Course summary

Estimated time

00:15

Overview

This unit summarizes the course.

Unit objectives

- Explain how the course met its learning objectives
- Access the IBM Training website
- Identify other IBM Training courses that are related to this topic
- Locate appropriate resources for further study

Course summary

© Copyright IBM Corporation 2017

Figure 4-1. Unit objectives

Course objectives

- Describe the AAA framework within the IBM DataPower Gateway
- Explain the purpose of each step in an access control policy
- Configure a AAA action to enforce authentication and authorization policies that are in a AAA information file and in an LDAP server
- Describe the OAuth 2.0 framework
- Explain the role that a DataPower gateway performs in an OAuth 2.0 framework
- Configure the DataPower objects that are used for OAuth 2.0 interactions
- Define Social Login
- Describe how to configure Social Login in DataPower
- Configure an OIDC client

[Course summary](#)

© Copyright IBM Corporation 2017

Figure 4-2. Course objectives

Lab exercise solutions

- Solutions are available in the `solution` subdirectory:

`<lab_files>/Solutions`

- Remember to change
 - Port numbers
 - Back-end server (**Network > Interface > DNS Settings > Static Hosts**)
 - Front IP addresses (**Network > Interface > Host Alias**)

To learn more on the subject

- IBM Training website:
<http://www.ibm.com/training>
- Webcast: How to define developer resources in DataPower Virtual Edition for Developers v7
<http://youtu.be/EaQyQWwQVIY>
- Webcast: VW750, Technical Introduction to IBM WebSphere DataPower Gateway Appliance V7.5.0
<https://youtu.be/yYk5Bzuie4g>
https://mediacenter.ibm.com/media/t/1_fb2tsml1
- DataPower V7.5 documentation in the IBM Knowledge Center
http://www.ibm.com/support/knowledgecenter/SS9H2Y_7.5.0
- IBM Redbooks:
<http://www.redbooks.ibm.com>
Search on “DataPower”
- developerWorks articles:
<http://www.ibm.com/developerworks/>
Search on “DataPower”

Course summary

© Copyright IBM Corporation 2017

Figure 4-4. To learn more on the subject

Enhance your learning with IBM resources

Keep your IBM Cloud skills up-to-date

- IBM offers resources for:
 - Product information
 - Training and certification
 - Documentation
 - Support
 - Technical information



- To learn more, see the IBM Cloud Education Resource Guide:
 - www.ibm.biz/CloudEduResources

Course summary

© Copyright IBM Corporation 2017

Figure 4-5. Enhance your learning with IBM resources

Unit summary

- Explain how the course met its learning objectives
- Access the IBM Training website
- Identify other IBM Training courses that are related to this topic
- Locate appropriate resources for further study

Course summary

© Copyright IBM Corporation 2017

Figure 4-6. Unit summary

Course completion



You have completed this course:

AAA, OAuth, and OIDC in IBM DataPower V7.5

Do you have any questions?

[Course summary](#)

Figure 4-7. Course completion

© Copyright IBM Corporation 2017

Appendix A. List of abbreviations

A

AAA	authentication, authorization, and auditing
AAD	additional authenticated data
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
APAR	authorized program analysis report
API	application programming interface
ASCII	American Standard Code for Information Interchange

B

BPM	business process management
------------	-----------------------------

C

CBA	context-based access
CEK	content encryption key
CLI	command-line interface
CRT	Chinese Remainder Theorem

D

DAP	Directory Access Protocol
DB	database
DER	Distinguished Encoding Rules
DH	Diffie-Hellman
DN	distinguished name
DNS	Dynamic Name Server

E

EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ECMA	European Computer Manufacturers Association

EP enforcement point

F

FLWOR for, let, where, order by, return

FSH front side handler

G

GB gigabyte

GCM Galois/Counter Mode

GUI graphical user interface

H

HMAC hash message authentication code

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

HTTPS HTTP over SSL

I

IANA Internet Assigned Numbers Authority

IBM International Business Machines Corporation

IETF Internet Engineering Task Force

IP Internet Protocol

IV initialization vector

J

J2SE Java Platform, Standard Edition

JDBC Java Database Connectivity

JMS Java Message Service

JOSE JSON Object Signing and Encryption

JSON JavaScript Object Notation

JWA JSON Web Algorithm

JWE JSON Web Encryption

JWK JSON Web Key

JWS JSON Web Signature

JWT JSON Web Token

K

KDF key derivation function

L

LDAP Lightweight Directory Access Protocol

LDIF LDAP Data Interchange Format

LTPA Lightweight Third Party Authentication

M

MAC message authentication code

MFA message filter action

MFA multi-factor authentication

MIME Multipurpose Internet Mail Extensions

MPGW multi-protocol gateway

N

npm node package manager

NSS network security services

O

OAEP Optimal Asymmetric Encryption Padding

OASIS Organization for the Advancement of Structured Information Standards

OAuth Open standard for Authorization

OIDC OpenID Connect

OP OpenID Provider

OTP one-time password

P

PBES Password Based Encryption Scheme

PCRE Perl Compatible Regular Expressions

PDF Portable Document Format

PEM Privacy Enhanced Mail

PI processing instruction

PKCS Public Key Cryptography Standard

PKI public key infrastructure

PKIX Public Key Infrastructure for X.509 Certificates (IETF)

POX	plain old XML
PS	Probabilistic Signature Scheme
R	
REST	Representational State Transfer
RFC	Request for Comments
RP	Relying Party
RPC	Remote Procedure Call
RSA	Rational Software Architect
RSS	Really Simple Syndication
S	
SAF	System Authorization Facility
SAML	Security Assertion Markup Language
SDK	software development kit
SHA	secure hash algorithm
SLM	service level management
SLM	service level monitoring
SNMP	Simple Network Management Protocol
SOA	service-oriented architecture
SOAP	Usage note: SOAP is not an acronym; it is a word in itself (formerly an acronym for Simple Object Access Protocol)
SPNEGO	Simple and Protected GSSAPI Negotiation Mechanism
SPVC	self-paced virtual classroom
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Sockets Layer
SSO	single sign-on
STS	Security Token Service
T	
TLS	Transport Layer Security
U	
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

USB	Universal Serial Bus
UTC	Coordinated Universal Time
W	
WS	web services
WSDL	Web Services Description Language
WSP	web service proxy
WTS	web token service
WWW	World Wide Web
X	
XACML	Extensible Access Control Markup Language
XML	Extensible Markup Language
XMLFW	XML firewall
XPath	XML Path Language
XSD	XML Schema Definition
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformation



IBM Training

