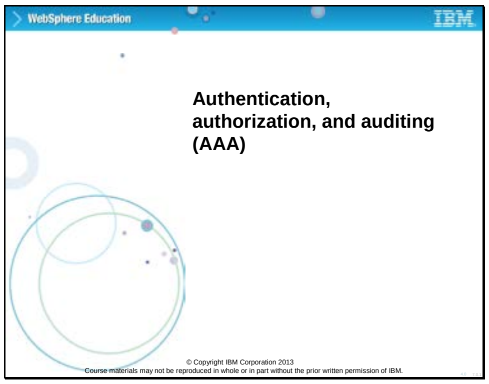
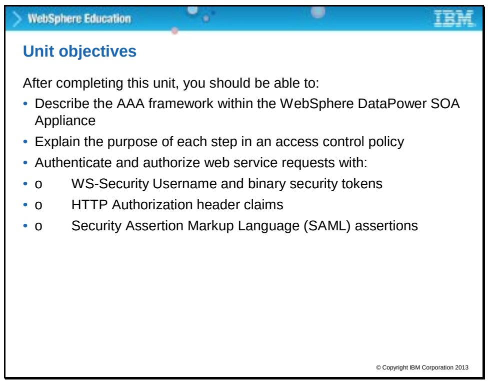
Slide 1



WebSphere Education

Authentication, authorization, and auditing (AAA)

Slide 2



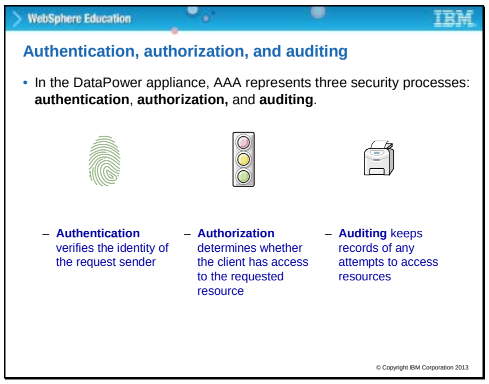
WebSphere Education

Unit objectives

After completing this unit, you should be able to:

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

Slide 3


WebSphere Education

IBM

Authentication, authorization, and auditing

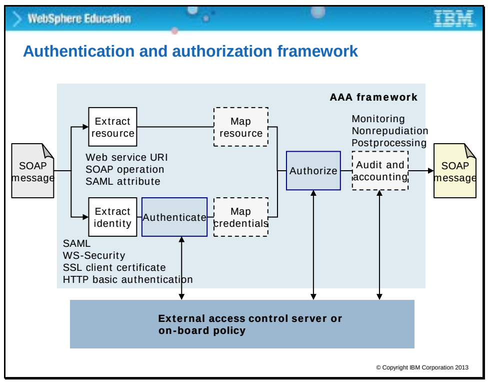
- In the DataPower appliance, AAA represents three security processes: **authentication**, **authorization,** and **auditing**.
  - **Authentication** verifies the identity of the request sender
  - **Authorization** determines whether the client has access to the requested resource
  - **Auditing** keeps records of any attempts to access resources

Authentication, authorization, and auditing

In the DataPower appliance, 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.

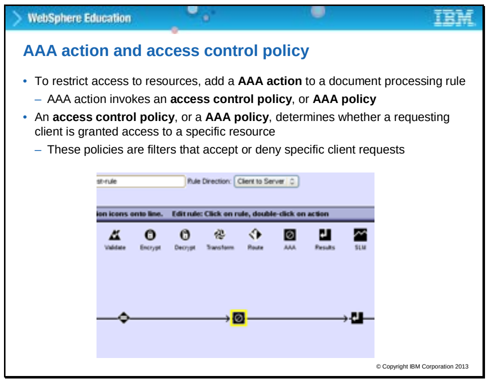
Slide 4



Authentication and authorization framework

This diagram shows the high-level flow of a message through the Triple-A process. First, extract the credentials from the incoming message. These credentials are authenticated against an internal or external list. Next, extract the resources that are trying to be accessed. Then, check authorization by comparing the authenticated user credentials against the resources again by consulting an internal or external list. At each of these stages, some transformation of the credentials or resources might take place to satisfy the needs of the destination system. Finally, opt to maintain an audit trail of what happened, which is the third A in the acronym. Also, do some post-processing before sending the message out. The processing can include transformation, encryption, digital signatures, and other forms of message manipulation.

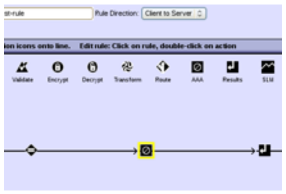
Slide 5



WebSphere Education

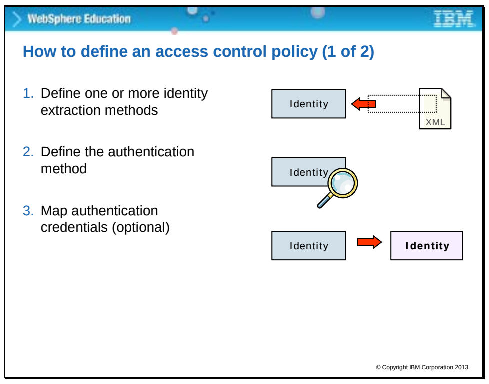
AAA action and access control policy

- 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


st-rule
Rule Direction: Client to Server
ion icons onto line. Edit rule: Click on rule, double-click on action
Validate
Encrypt
Decrypt
Transform
Route
AAA
Results
SLM

AAA action and access control policy
The usual way of implementing a Triple-A policy is to use a Triple-A action on a rule. By double-clicking the Triple-A icon, you can set all the parameters that it needs. The next few slides go through those parameters.

Slide 6


WebSphere Education

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

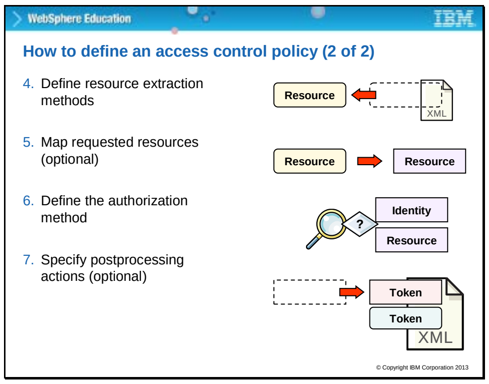
1. Define one or more identity extraction methods
2. Define the authentication method
3. Map authentication credentials (optional)

Identity
XML
Identity
Identity
Identity

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

Here is the first part of how to define an access control policy. First, define one or more identity extraction methods – look as some of the choices on a subsequent slide. Next, define the used authentication methods, and then optionally map the authentication credentials.

Slide 7


WebSphere Education

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

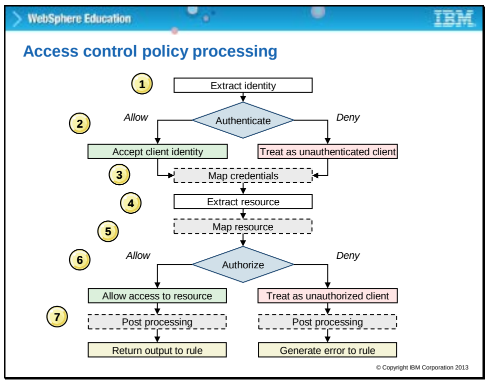
4. Define resource extraction methods
5. Map requested resources (optional)
6. Define the authorization method
7. Specify postprocessing actions (optional)

Resource
XML
Resource
Resource
Identity
?
Resource
Token
Token
XML

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

Continuing from the previous slide, define the resource extraction used methods; then optionally map the requested resources. Next, define the authorization method that is used, and finally specify any post-processing actions, if any.

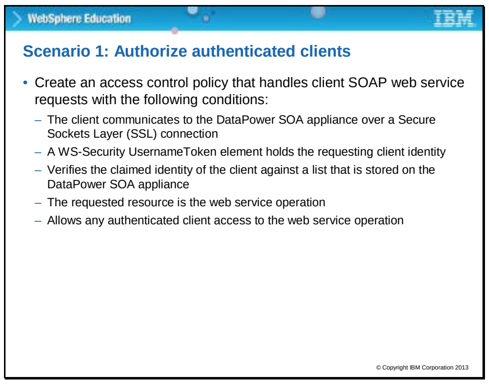
Slide 8



Access control policy processing

This diagram shows the decisions that are made during normal Triple-A processing. First, decide whether the user is valid. If not, you might choose to reject the message, or assign the authentication to a guest user account. Next, check the authorization of the account, which might lead to a valid or invalid outcome. If it is valid, proceed to post-processing and send the message out intact. If it is invalid, optionally do some error processing, and terminate processing the message.

Slide 9


WebSphere Education

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 SOA appliance 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 SOA appliance
  - The requested resource is the web service operation
  - Allows any authenticated client access to the web service operation

Scenario 1: Authorize authenticated clients

Now look at some typical triple-A scenarios. For first one, create an access control policy that handles client SOAP web service requests with the following conditions:

The client communicates to the DataPower SOA appliance over a Secure Sockets Layer (SSL) connection.

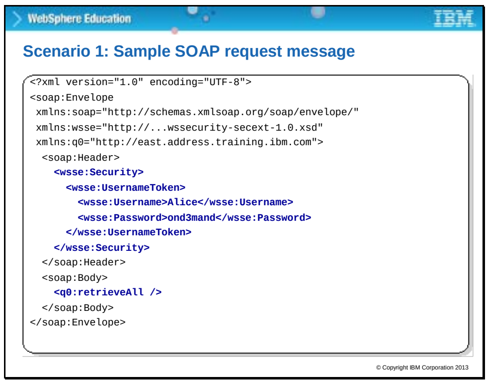
A WS-Security UsernameToken element holds the requesting client identity.

Verify the claimed identity of the client against a list that is stored on the DataPower SOA appliance.

The requested resource is the web service operation.

The policy allows any authenticated client access to the web service operation.

Slide 10



WebSphere Education

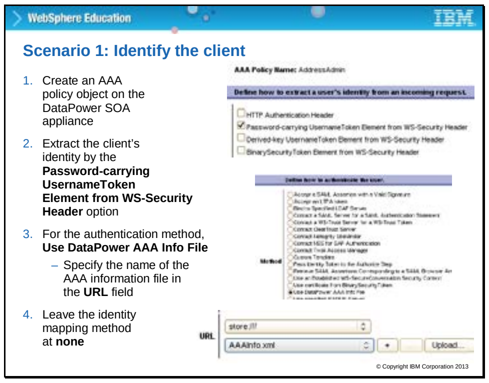
Scenario 1: Sample SOAP request message

```
<?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>
```

Scenario 1: Sample SOAP request message

The example shows a simple Web Services Security header with a name and password in clear text. Because of clear text, it is important that the protocol is HTTPS to ensure that the sensitive data is encrypted.

Slide 11



Scenario 1: Identify the client

1. Create an AAA policy object on the DataPower SOA appliance
2. Extract the client's identity by the **Password-carrying UsernameToken Element from WS-Security Header** option
3. For the authentication method, **Use DataPower AAA Info File**
   – Specify the name of the AAA information file in the **URL** field
4. Leave the identity mapping method at **none**

AAA Policy Name: AddressAdmin

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

Define how to authenticate the user.

Method

- Accept a SAML Assertion with a Valid Signature
- Accept an LTPA token
- Bind to Specified LDAP Server
- Contact a SAML Server for a SAML Authentication Statement
- Contact a WS-Trust Server for a WS-Trust Token
- Contact ClearTrust Server
- Contact Netegrity SiteMinder
- Contact NSS for SAF Authentication
- Contact Tivoli Access Manager
- Custom Template
- Pass Identity Token to the Authorize Step
- Retrieve SAML Assertions Corresponding to a SAML Browser Ar
- Use an Established WS-SecureConversation Security Context
- Use certificate from BinarySecurityToken
- ● Use DataPower AAA Info File

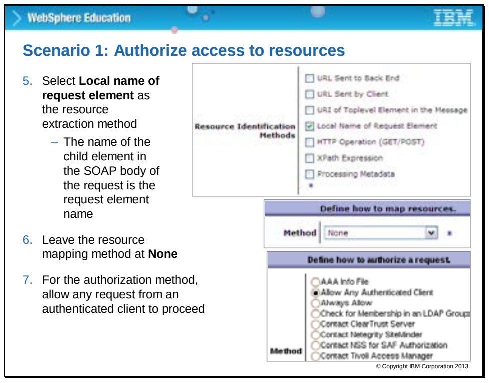
URL store:/// AAAInfo.xml

Scenario 1: Identify the client

Here is how to identify the client. Create a AAA policy object in the policy rule to extract the clients identity by using the Password-carrying UsernameToken Element from the WS-Security Header.

For the authentication method, use an internally stored AAA Info File. Specify the name of the AAA information file in the URL field, and leave the identity mapping method set at “none”.

Slide 12



WebSphere Education

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

Resource Identification Methods:
- [ ] URL Sent to Back End
- [ ] URL Sent by Client
- [ ] URI of Toplevel Element in the Message
- [✓] Local Name of Request Element
- [ ] HTTP Operation (GET/POST)
- [ ] XPath Expression
- [ ] Processing Metadata

Define how to map resources.

Method: None

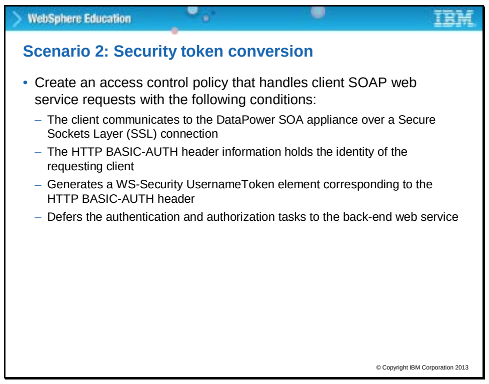
Define how to authorize a request.

Method:
- AAA Info File
- [✓ Allow Any Authenticated Client]
- Always Allow
- Check for Membership in an LDAP Group
- Contact ClearTrust Server
- Contact Netegrity SiteMinder
- Contact NSS for SAF Authorization
- Contact Tivoli Access Manager

Scenario 1: Authorize access to resources

Next, authorize access to resources. Authorization is done by selecting the Local name of the 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. Leave the resource mapping method at None; then, for the authorization method. Allow any request from an authenticated client to proceed.

Slide 13



WebSphere Education

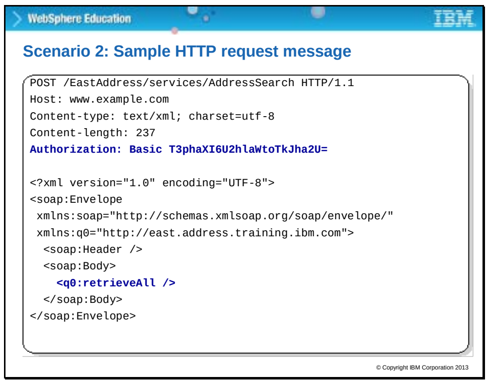
Scenario 2: Security token conversion

- Create an access control policy that handles client SOAP web service requests with the following conditions:
  - The client communicates to the DataPower SOA appliance 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

Scenario 2: Security token conversion

For the second scenario, look at doing a 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 SOA appliance over a Secure Sockets Layer (SSL) connection, sending an HTTP document with a BASIC-AUTH header that contains the identity of the requesting client. Generate a WS-Security UsernameToken element corresponding to the HTTP BASIC-AUTH header, and defer the authentication and authorization tasks to the back-end web service.

Slide 14


WebSphere Education

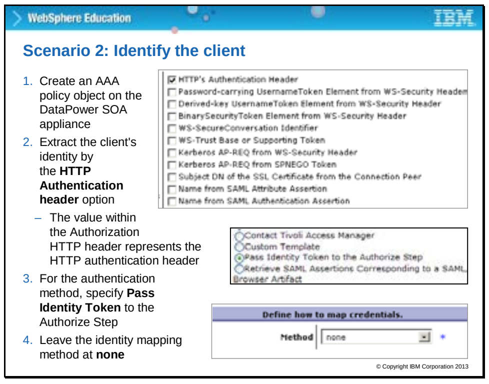
Scenario 2: Sample HTTP request message

```
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>
```

Scenario 2: Sample HTTP request message
In this example, a Basic-Auth header contains a tokenized version of the user identity. Because it is encrypted, it does not require use of an encryption protocol in this case.

Slide 15


WebSphere Education

Scenario 2: Identify the client

1. Create an AAA policy object on the DataPower SOA appliance
2. Extract the client's identity by 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 the Authorize Step
4. Leave the identity mapping method at **none**

- [✓] HTTP's 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 the SSL Certificate from the Connection Peer
- [ ] Name from SAML Attribute Assertion
- [ ] Name from SAML Authentication Assertion

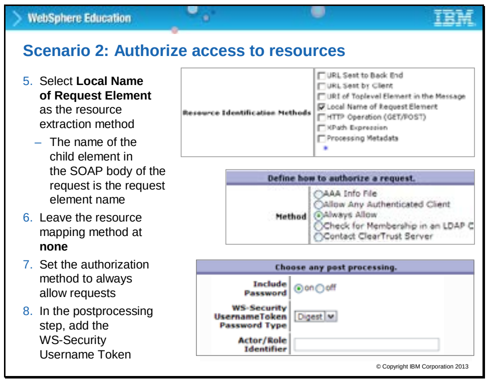
- ○ Contact Tivoli Access Manager
- ○ Custom Template
- ● Pass Identity Token to the Authorize Step
- ○ Retrieve SAML Assertions Corresponding to a SAML Browser Artifact

Define how to map credentials.

Method: none *

Scenario 2: Identify the client
To implement this scenario, first identify the client. Start by creating a AAA policy object in a policy rule that extracts the clients identity by using the HTTPs Authentication header option.
The value within the Authorization HTTP header represents the HTTP authentication header. For the authentication method, specify Pass Identity Token to the Authorize Step, and leave the identity mapping method at none.

Slide 16



WebSphere Education

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

Resource Identification Methods
- [ ] URL Sent to Back End
- [ ] URL Sent by Client
- [ ] URI of Toplevel Element in the Message
- [✓] Local Name of Request Element
- [ ] HTTP Operation (GET/POST)
- [ ] XPath Expression
- [ ] Processing Metadata

Define how to authorize a request.

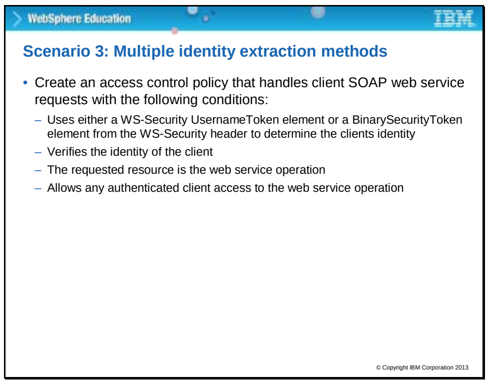
Method	AAA Info File / Allow Any Authenticated Client / [✓ Always Allow] / Check for Membership in an LDAP C / Contact ClearTrust Server

Choose any post processing.

Include Password	[✓ on] off
WS-Security UsernameToken Password Type	Digest
Actor/Role Identifier	

Scenario 2: Authorize access to resources
To authorize access to the resources, select Local name of request element as the resource extraction method that uses the name of the child element in the SOAP body as the request element name. Leave the resource mapping method at none and set the authorization method to Always Allow requests. In the post processing step, choose “Add WS-Security Username Token”.

Slide 17



WebSphere Education

IBM

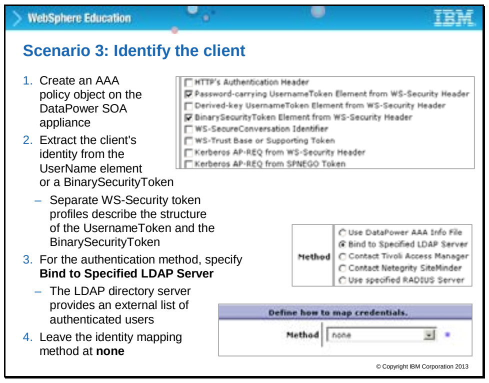
Scenario 3: Multiple identity extraction methods

- 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 clients 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

Scenario 3: Multiple identity extraction methods

In the third scenario, multiple identity extraction methods are shown. First, create an access control policy that handles client SOAP web service requests with the following conditions. It uses either a WS-Security UsernameToken element or a BinarySecurityToken element from the WS-Security header to determine the clients identity. Regardless of which it is, verify the identity of the client. Then, get the name of the requested resource, which is the web service operation, and allow any authenticated client access to that operation.

Slide 18



WebSphere Education

Scenario 3: Identify the client

1. Create an AAA policy object on the DataPower SOA appliance
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 Specified LDAP Server**
   – The LDAP directory server provides an external list of authenticated users
4. Leave the identity mapping method at **none**

- ☐ HTTP's 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

Method: Use DataPower AAA Info File [✓ Bind to Specified LDAP Server] Contact Tivoli Access Manager Contact Netegrity SiteMinder Use specified RADIUS Server

Define how to map credentials.

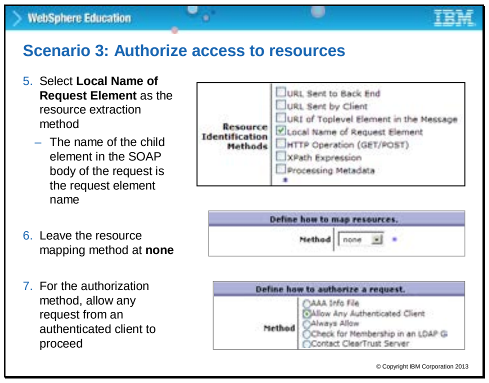
Method: none

© Copyright IBM Corporation 2013

Scenario 3: Identify the client

First, identify the client. As usual, create a AAA action on the policy rule, and extract the clients identity from the UserName element or a BinarySecurityToken. Remember, the identity extraction method choices are presented as check-boxes, allowing one to choose more than one at a time. The example is allowing for separate WS-Security token profiles that describe the structure of the UsernameToken and the BinarySecurityToken. For the authentication method, in this case specify “Bind to Specified LDAP Server” which provides you with an external list of authenticated users. When again, leave the identity mapping method at none.

Slide 19



WebSphere Education

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
- ☐ URL Sent to Back End
- ☐ URL Sent by Client
- ☐ URI of Toplevel Element in the Message
- ✓ Local Name of Request Element
- ☐ HTTP Operation (GET/POST)
- ☐ XPath Expression
- ☐ Processing Metadata

Define how to map resources.

Method: none

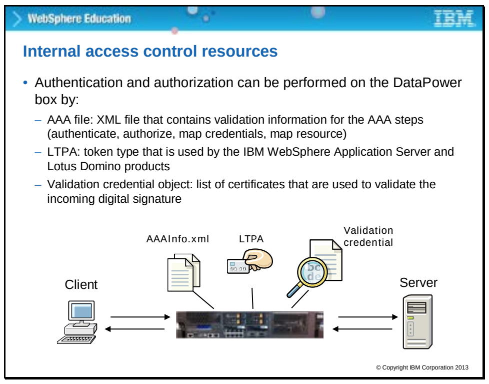
Define how to authorize a request.

Method:
- AAA Info File
- [✓ Allow Any Authenticated Client]
- Always Allow
- Check for Membership in an LDAP G
- Contact ClearTrust Server

Scenario 3: Authorize access to resources

Now, authorize access to resources by choosing the method of identifying the resource. When again, choose to use the Local name of request element as the resource extraction method. As before, the name of the child element in the SOAP body of the request is the request element name. Continue to leave the resource mapping method at none, and again allow any request from an authenticated client to proceed.

Slide 20


WebSphere Education

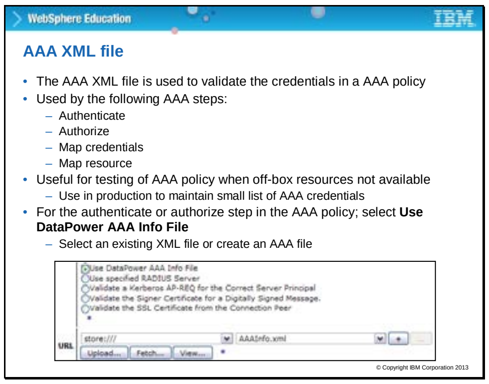
Internal access control resources

- 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 is used by the IBM WebSphere Application Server and Lotus Domino products
  - Validation credential object: list of certificates that are used to validate the incoming digital signature

© Copyright IBM Corporation 2013

Internal access control resources
You listened to talk about by using internally stored access control mechanisms, and have seen how to use the AAAinfo.xml file. In addition to using that file for authentication and authorization, also use LTPA, which is a Token type that is used by the IBM WebSphere Application Server and Lotus Domino products. And a third internal mechanism is the use of a validation credential object, being a list of certificates that are used to validate the incoming digital signature. These three examples show how authentication can take place within the DataPower box with no necessity to reference external sources. Obviously, the process can be much more efficient than having to contact remote servers.

Slide 21



WebSphere Education

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 DataPower AAA Info File**
  - Select an existing XML file or create an AAA file

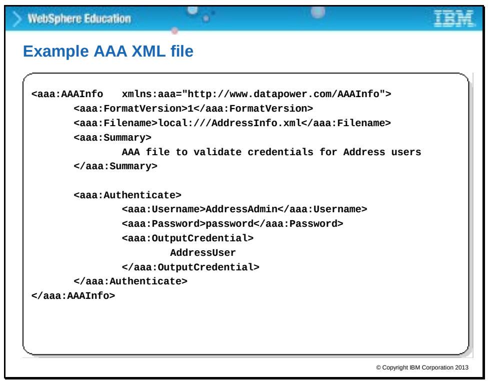
[✓ Use DataPower AAA Info File]
Use specified RADIUS Server
Validate a Kerberos AP-REQ for the Correct Server Principal
Validate the Signer Certificate for a Digitally Signed Message.
Validate the SSL Certificate from the Connection Peer

URL: store:/// AAAInfo.xml

AAA XML file

The AAA XML file is used to validate the credentials in a AAA policy in the following steps – Authenticate, Authorize, Map credentials, and Map resource.
It is useful for the testing of a AAA policy when off-box resources are not available. However, its use in production is not suggested unless you are by using it only to maintain small list of AAA credentials. To use the AAAinfo.xml file, in the authenticate or authorize step in the AAA policy, select “Use DataPower AAA Info File” option, then select an existing XML file or create a AAA file.
The AAAinfo.xml file is obviously an editable text file, and therefore is easy to maintain if it contains a limited number of entries. Its usefulness, however, is limited to situations where you have few relatively static accounts to maintain. In a production environment, where you might potentially have tens of thousands, or even hundreds of thousands of customers who are creating, modifying, and deleting their account information on an ad hoc basis. It would obviously be impossible to maintain all of this dynamic information in a simple text file of this nature. It is highly suggested that the AAAinfo.xml file is used in a development and test environment.

Slide 22



WebSphere Education

Example AAA XML file

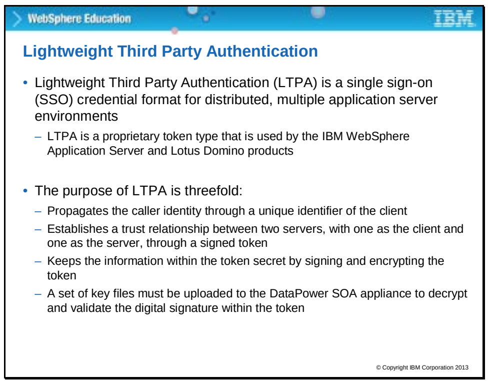
```
<aaa:AAAInfo   xmlns:aaa="http://www.datapower.com/AAAInfo">
       <aaa:FormatVersion>1</aaa:FormatVersion>
       <aaa:Filename>local:///AddressInfo.xml</aaa:Filename>
       <aaa:Summary>
               AAA file to validate credentials for Address users
       </aaa:Summary>

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

Example AAA XML file

Here is an example of the AAAinfo.xml file. As you can see, it has a relatively simple structure, and you can easily edit it to include credentials for testing purposes.

Slide 23


WebSphere Education

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 is used by the IBM WebSphere Application Server and Lotus Domino products
- 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 SOA appliance to decrypt and validate the digital signature within the token

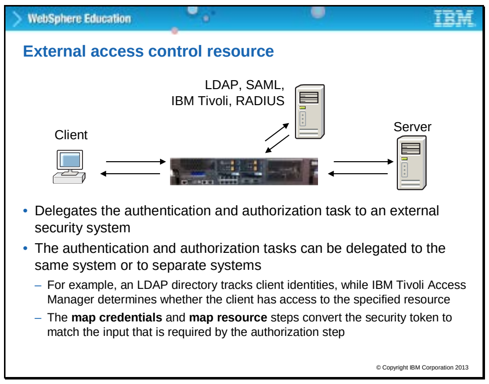
Lightweight Third Party Authentication

You heard LTPA mentioned earlier – it stands for Lightweight Third Party Authentication and is a single sign-on (SSO) credential format for distributed, multiple application server environments.

LTPA is a proprietary token type that is used by the IBM WebSphere Application Server and Lotus Domino products.

The purpose of LTPA is threefold. It propagates the caller identity through a unique identifier of the client. It establishes a trust relationship between two servers, with one as the client and one as the server, through a signed token. And it keeps the information within the token secret by signing and encrypting the token. A set of key files must be uploaded to the DataPower SOA appliance to decrypt and validate the digital signature within the token.

Slide 24


WebSphere Education
IBM

External access control resource

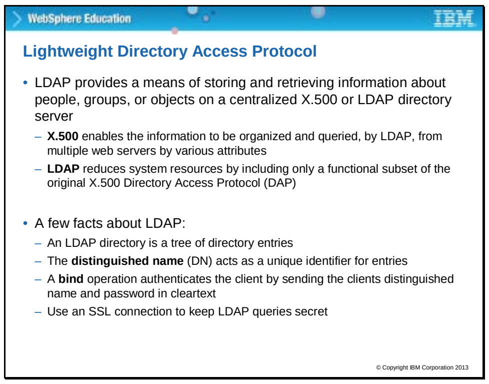
LDAP, SAML,
IBM Tivoli, RADIUS
Client
Server

- 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 Tivoli 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 is required by the authorization step

© Copyright IBM Corporation 2013

External access control resource
In addition to the three internal methods that are mentioned, you also can use external access control resources.
You can delegate the authentication and authorization task to an external security system. And you can choose to use the same service for both the authentication and authorization tasks, or you can split it up and use separate systems for each task.
For example, you might use an LDAP directory to track client identities while an IBM Tivoli 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 is required by the authorization step.
DataPower supports various servers, including Tivoli Access Manager, Radius, SPNEGO, LDAP, Microsoft Active Directory, and many others. Since consulting external directories is a relatively time-consuming process, the option exists to cache certain data for rapid access. A more efficient methodology involves the use of tokens. More about token use later.

Slide 25



WebSphere Education

Lightweight Directory Access Protocol

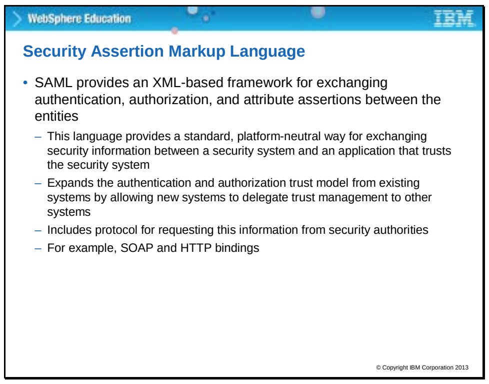
- LDAP provides a means of storing and retrieving information about people, groups, or objects on a centralized X.500 or LDAP directory server
  - **X.500** enables the information to be organized and queried, by LDAP, from multiple web servers by various attributes
  - **LDAP** reduces system resources 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) acts as a unique identifier for entries
  - A **bind** operation authenticates the client by sending the clients distinguished name and password in cleartext
  - Use an SSL connection to keep LDAP queries secret

Lightweight Directory Access Protocol

Lightweight Directory Access Protocol, commonly referred to by its acronym “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 using LDAP, from multiple web servers by using 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 – its directory is a tree of directory entries. The distinguished name (DN) acts as a unique identifier for entries. A bind operation authenticates the client by sending the clients distinguished name and password in clear text, so it is advisable to use an SSL connection to keep LDAP queries secret.

Slide 26



WebSphere Education

IBM

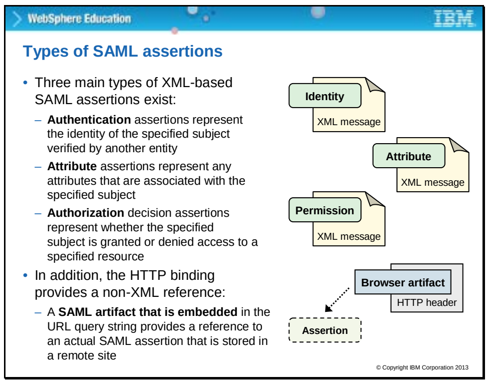
Security Assertion Markup Language

- SAML provides an XML-based framework for exchanging authentication, authorization, and attribute assertions between the entities
  - This language 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

Security Assertion Markup Language

Security Assertion Markup Language, or “SAML”, provides an XML-based framework for exchanging authentication, authorization, and attribute assertions between entities. It provides a standard, platform-neutral way for exchanging security information between a security system and an application that trusts the security system. It expands the authentication and authorization trust model from existing systems by allowing new systems to delegate trust management to other systems. It also includes a protocol for requesting this information from security authorities, such as SOAP and HTTP bindings.

Slide 27



WebSphere Education

IBM

Types of SAML assertions

- Three main types of XML-based SAML assertions exist:
  - **Authentication** assertions represent the identity of the specified subject verified by another entity
  - **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

Types of SAML assertions

Three main types of XML-based SAML assertions exist:

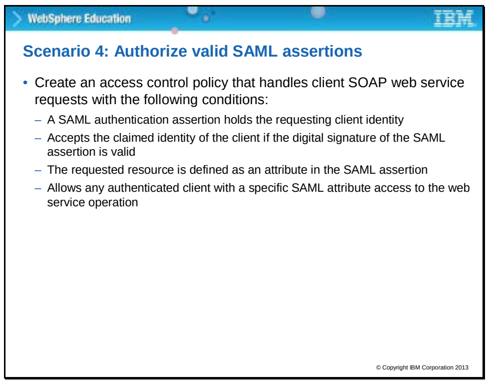
First, there are authentication assertions that represent the identity of the specified subject that another entity verifies.

Second, there are attribute assertions that represent any attributes that are associated with the specified subject.

And third, 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 stored in a remote site.

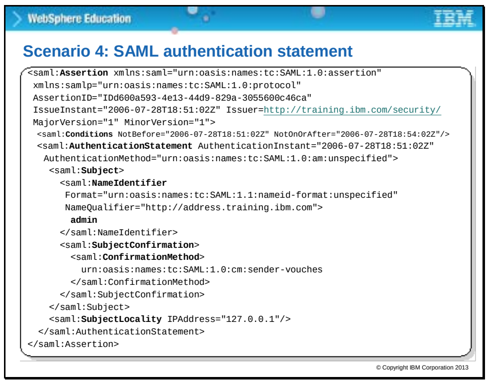
Slide 28


WebSphere Education

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

Scenario 4: Authorize valid SAML assertions
Continuing with the scenario examples – the fourth one shows how to 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.
The policy 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.
The policy then allows any authenticated client with a specific SAML attribute access to the web service operation.



WebSphere Education

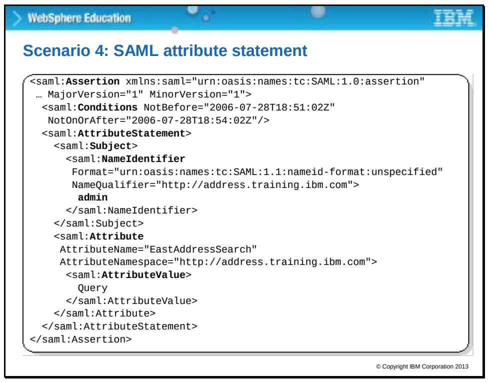
Scenario 4: SAML authentication statement

```
<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>
      <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>
```

Scenario 4: SAML authentication statement

Here is an example of a Security Assertion Markup Language header that is embedded in a SOAP header.

Slide 30



WebSphere Education

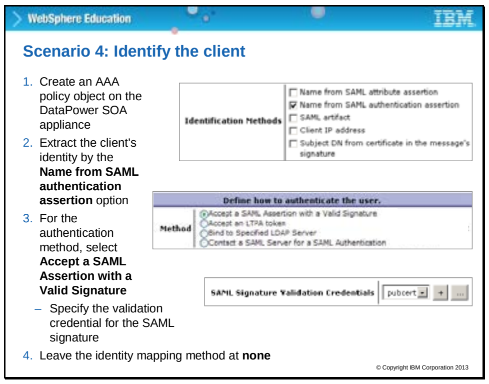
Scenario 4: SAML attribute statement

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

Scenario 4: SAML attribute statement

In addition to authentication and authorization information, SAML can be used to define attributes of the elements that are processed. These attributes provide a means of further identifying users and resources that are being processed. For example, a user whose name is "Fred," and an attribute of Fred might be "manager." Use this attribute of manager to provide further security checking.

Slide 31



WebSphere Education

Scenario 4: Identify the client

1. Create an AAA policy object on the DataPower SOA appliance
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 a Valid Signature**
   – Specify the validation credential for the SAML signature
4. Leave the identity mapping method at **none**

Identification Methods	☐ Name from SAML attribute assertion [✓ Name from SAML authentication assertion] ☐ SAML artifact ☐ Client IP address ☐ Subject DN from certificate in the message's signature

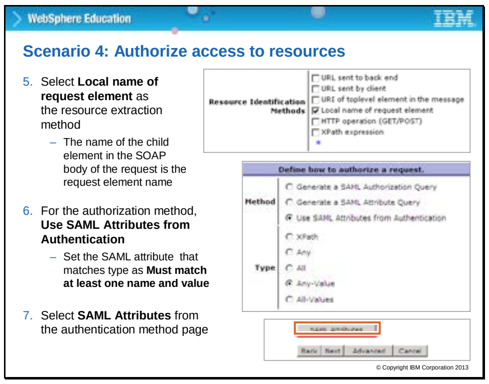
Define how to authenticate the user.

Method	[✓ Accept a SAML Assertion with a Valid Signature] ○ Accept an LTPA token ○ Bind to Specified LDAP Server ○ Contact a SAML Server for a SAML Authentication

SAML Signature Validation Credentials	pubcert

Scenario 4: Identify the client
Continuing with the fourth scenario, identify the client. As usual, first create a AAA policy object in the policy rule that extracts the clients identity by using the “name from SAML
Authentication assertion” option: For the authentication method, select “Accept a SAML Assertion with a Valid Signature” and specify the validation credential for the SAML signature – “pubcert” in this example. Finally, leave the identity mapping method at none.

Slide 32



WebSphere Education

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 **Must match at least one name and value**
7. Select **SAML Attributes** from the authentication method page

Resource Identification Methods
- [ ] URL sent to back end
- [ ] URL sent by client
- [ ] URI of toplevel element in the message
- [x] Local name of request element
- [ ] HTTP operation (GET/POST)
- [ ] XPath expression

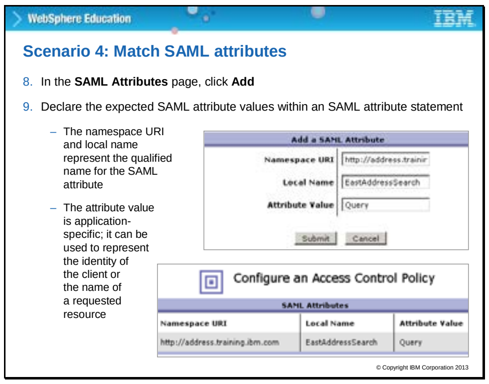
Define how to authorize a request.

Method: Generate a SAML Authorization Query / Generate a SAML Attribute Query / [✓ Use SAML Attributes from Authentication]

Type: XPath / Any / All / [✓ Any-Value] / All-Values

Scenario 4: Authorize access to resources
Next, want to authorize access to resources.
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.
For the authorization method, click “Use SAML Attributes from Authentication”, then set the SAML attribute that match type as “must match at least one name and value”.
Select the SAML Attributes from the authentication method page that is described on the next slide.

Slide 33


WebSphere Education

IBM

Scenario 4: Match SAML attributes

8. In 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	http://address.trainir
Local Name	EastAddressSearch
Attribute Value	Query

Submit Cancel

Configure an Access Control Policy

SAML Attributes

Namespace URI	Local Name	Attribute Value
http://address.training.ibm.com	EastAddressSearch	Query

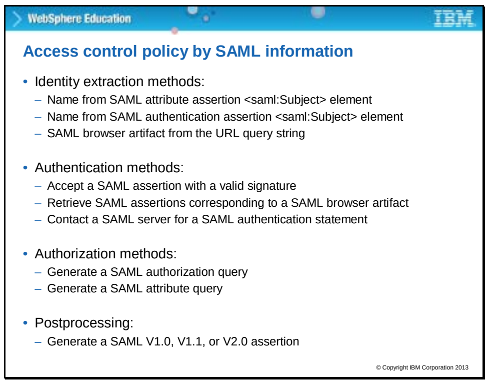
© Copyright IBM Corporation 2013

Scenario 4: Match SAML attributes

So here is how to match the SAML attributes.

In the SAML Attributes page, click Add. Declare the expected SAML attribute values within a 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.

Slide 34


WebSphere Education

Access control policy by SAML information

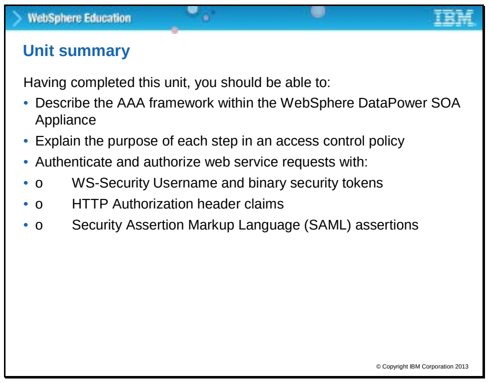
- Identity extraction methods:
  - Name from SAML attribute assertion <saml:Subject> element
  - Name from SAML authentication assertion <saml:Subject> element
  - SAML browser artifact from the URL query string
- Authentication methods:
  - Accept a SAML assertion with a valid signature
  - Retrieve 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

Access control policy by using SAML information

So, in summary, an access control policy by using SAML information consists of:

- Identity extraction methods that might use the name from a SAML attribute assertion <saml:Subject> element.
- Or the name from a SAML authentication assertion <saml:Subject> element.
- Or a SAML browser artifact from the URL query string.
- Authentication methods that might accept a SAML assertion with a valid signature or retrieve a SAML assertion corresponding to a SAML browser artifact, or might contact a SAML server for a SAML authentication statement.
- Authorization methods that might generate a SAML authorization query or a SAML attribute query.
- Post processing that can generate a SAML V1.0, V1.1 or V2.0 assertion.

Slide 35



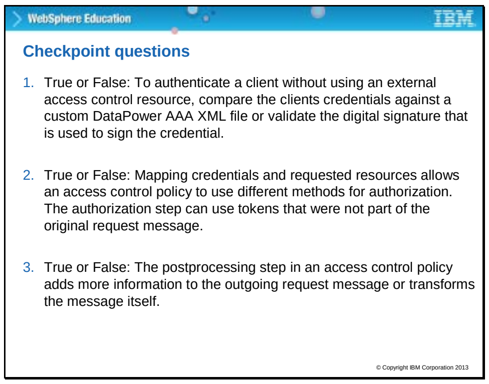
WebSphere Education

Unit summary

Having completed this unit, you should be able to:

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

Slide 36


WebSphere Education

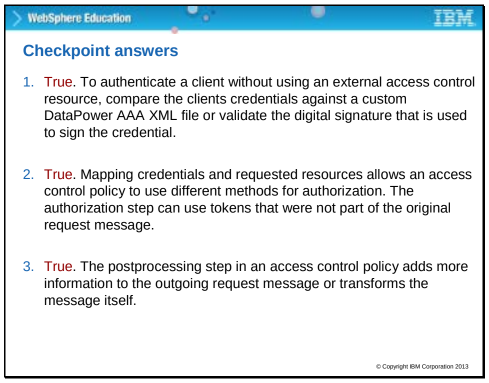
IBM

Checkpoint questions

1. True or False: To authenticate a client without using an external access control resource, compare the clients credentials against a custom DataPower AAA XML file or validate the digital signature that is used to sign the credential.
2. True or False: Mapping credentials and requested resources allows an access control policy to use different methods for authorization. The authorization step can use tokens that were not part of the original request message.
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.

.

Slide 37



WebSphere Education

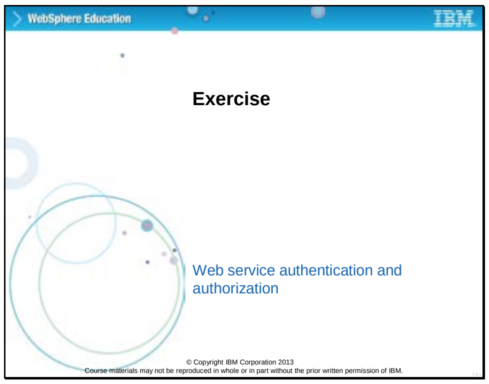
Checkpoint answers

1. True. To authenticate a client without using an external access control resource, compare the clients credentials against a custom DataPower AAA XML file or validate the digital signature that is used to sign the credential.

2. True. Mapping credentials and requested resources allows an access control policy to use different methods for authorization. The authorization step can use tokens that were not part of the original request message.

3. True. The postprocessing step in an access control policy adds more information to the outgoing request message or transforms the message itself.

Slide 38

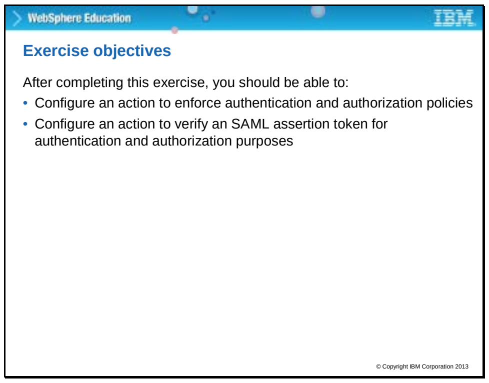


WebSphere Education

Exercise

Web service authentication and authorization

Slide 39



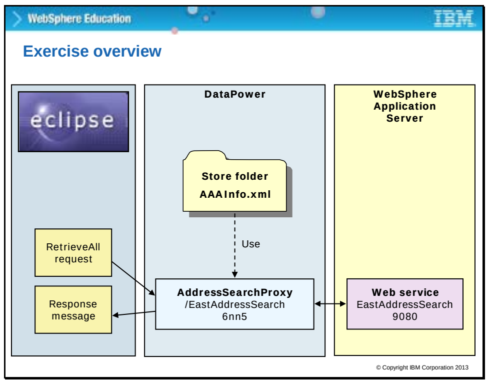
WebSphere Education

Exercise objectives

After completing this exercise, you should be able to:

- Configure an action to enforce authentication and authorization policies
- Configure an action to verify an SAML assertion token for authentication and authorization purposes

Slide 40



Exercise overview