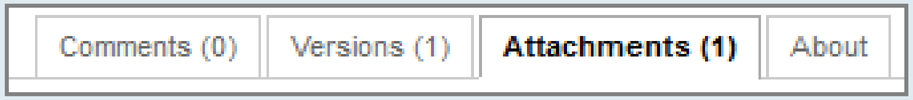
Authentication, authorization, and auditing (AAA)

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.



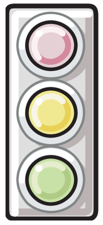
6. To save the file to your computer, click the document link and follow the prompts.

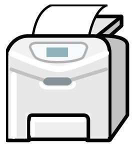
Unit objectives

- 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

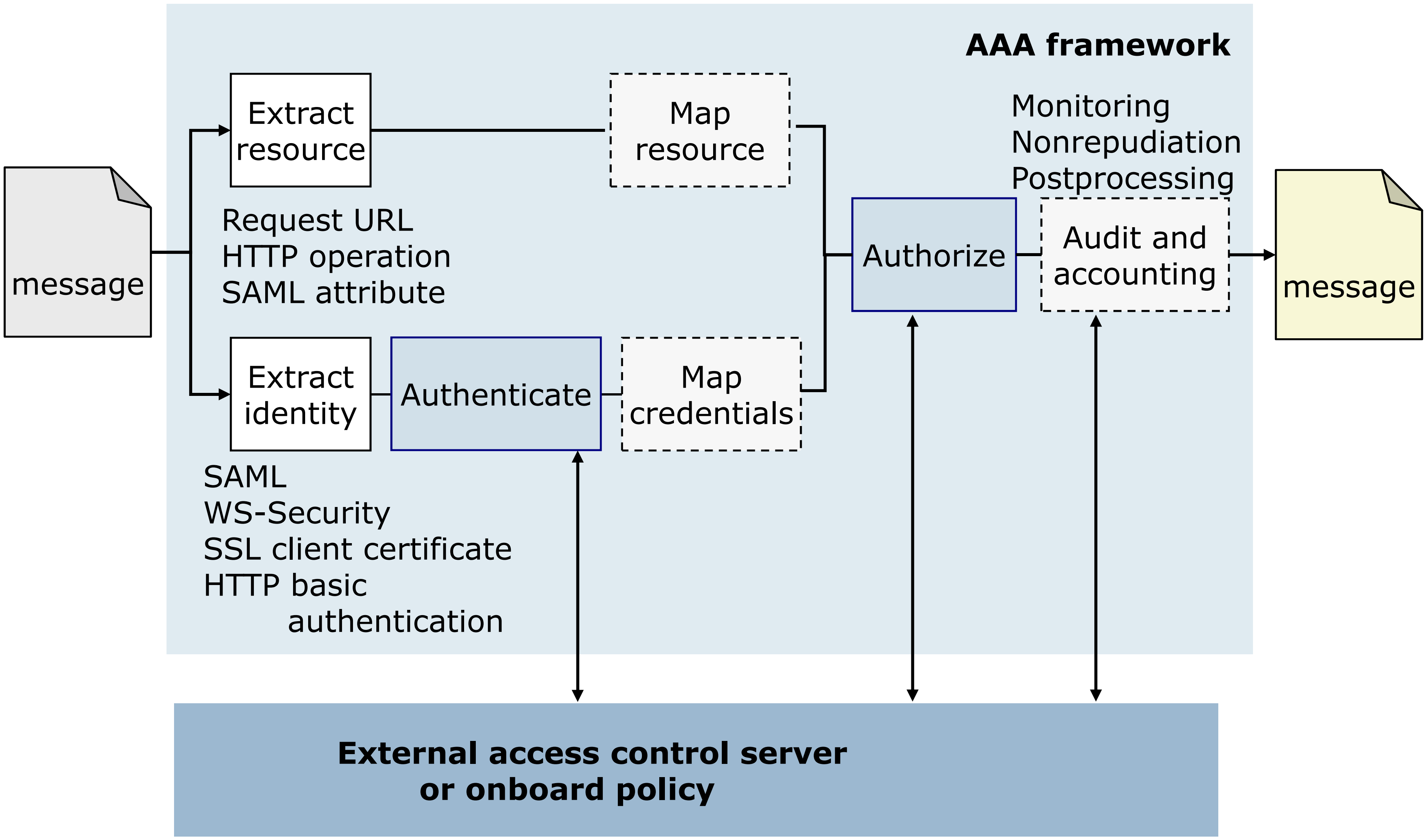
- In the DataPower gateway, AAA represents three security processes: **authentication**, **authorization**, and **auditing**





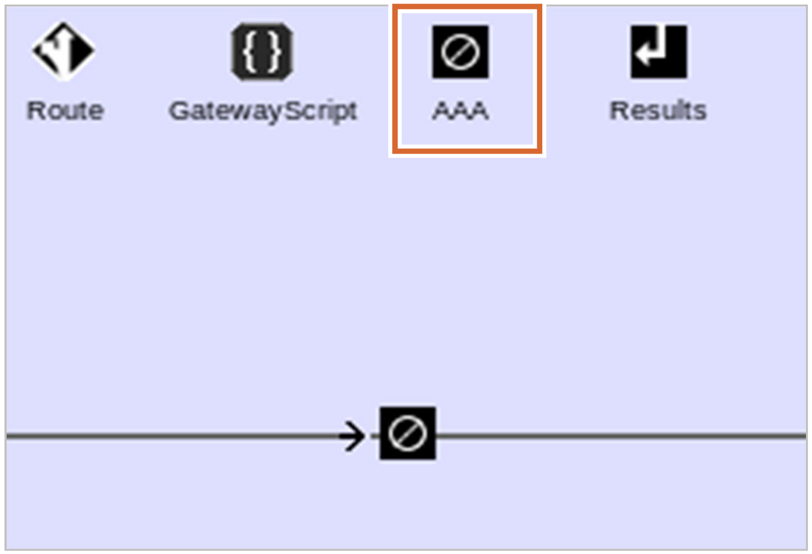
  - **Authentication** verifies the identity of the request sender
  - **Authorization** determines whether the client has access to the requested resource
  - **Auditing** keeps records of any attempts to access resources

Authentication and authorization framework



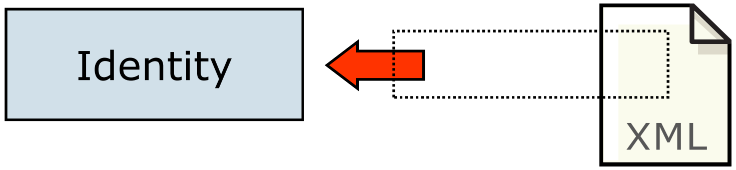
AAA action and access control policy

- 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

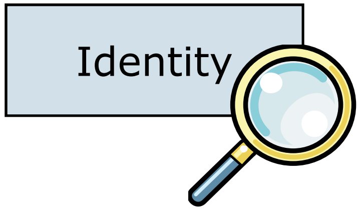


How to define an access control policy (1 of 2)

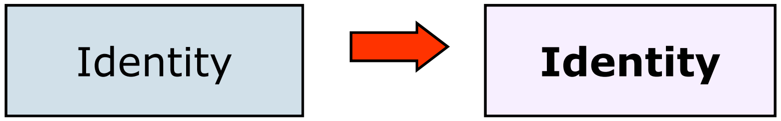
1. Define one or more identity extraction methods



2. Define the authentication method

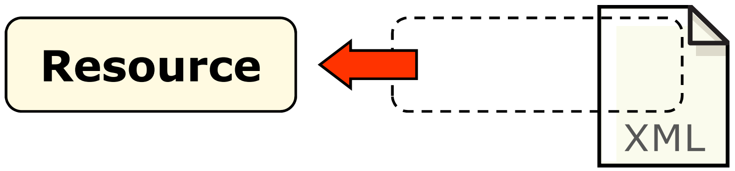


3. Map authentication credentials (optional)

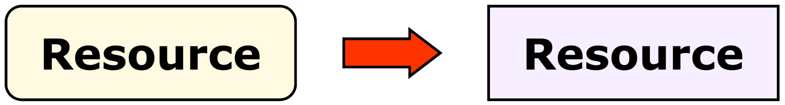


How to define an access control policy (2 of 2)

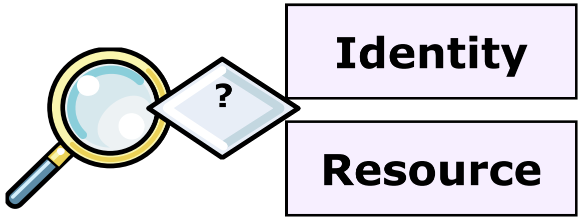
4. Define resource extraction methods



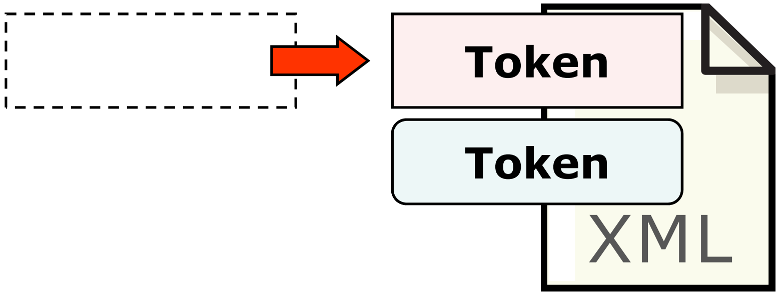
5. Map requested resources (optional)



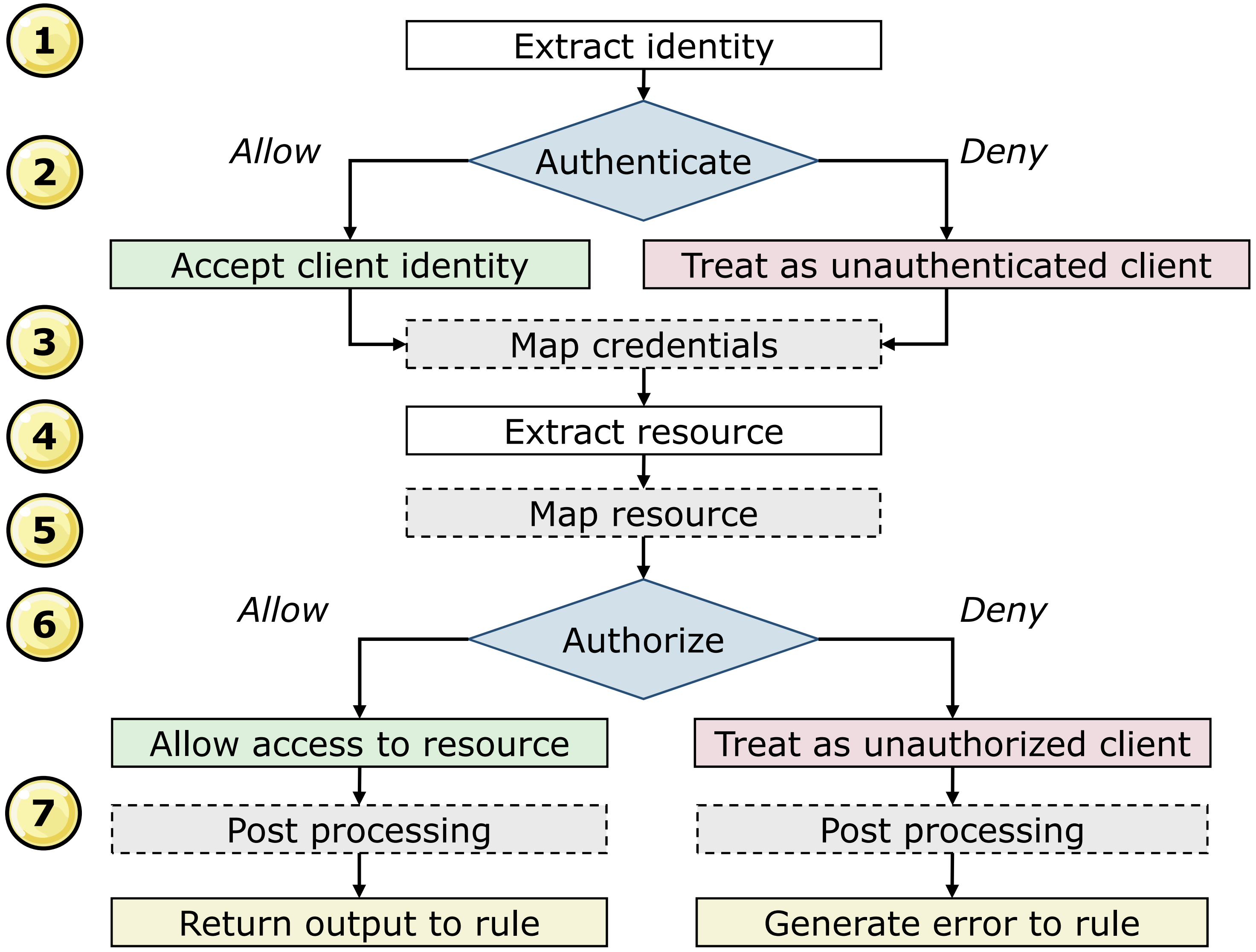
6. Define the authorization method



7. Specify postprocessing actions (optional)



Access control policy processing



Scenario 1: Authorize authenticated clients

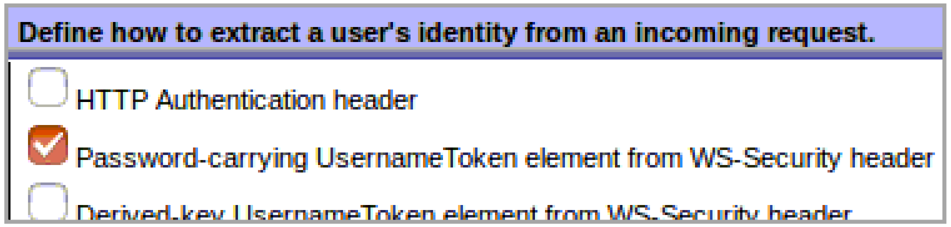
- Create an access control policy that handles client SOAP web service requests with the following conditions:
  - The client communicates to the DataPower gateway over a Secure Sockets Layer (SSL) connection
  - A WS-Security UsernameToken element holds the requesting client identity
  - Verifies the 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

Scenario 1: Sample SOAP request message

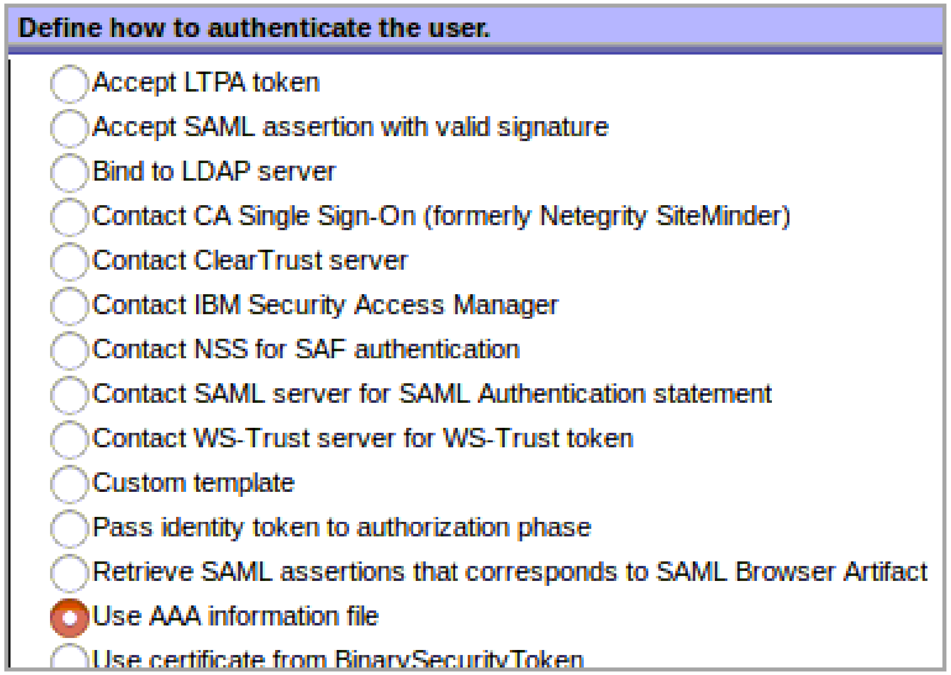
```
<?xml version="1.0" encoding="UTF-8">
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:wsse="http://...wssecurity-secext-1.0.xsd"
 xmlns:q0="http://east.address.training.ibm.com">
  <soap:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>Alice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

Scenario 1: Identify and authenticate the client

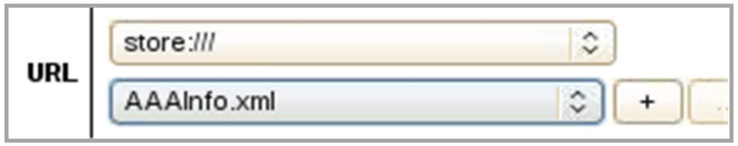
1. Create a AAA policy object on the DataPower gateway
2. Extract the client's identity with the **Password-carrying UsernameToken Element from WS-Security header** option
3. For the authentication method, **Use AAA information file**
   - Specify the name of the AAA information file in the **URL** field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

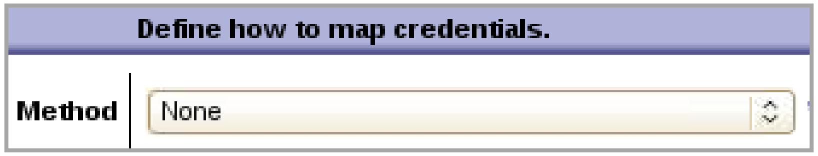

Define how to extract a user's identity from an incoming request.

- HTTP Authentication header
- [✓ Password-carrying UsernameToken element from WS-Security header]
- Derived-key UsernameToken element from WS-Security header


Define how to authenticate the user.

- 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


URL: store:/// AAAInfo.xml


Define how to map credentials.

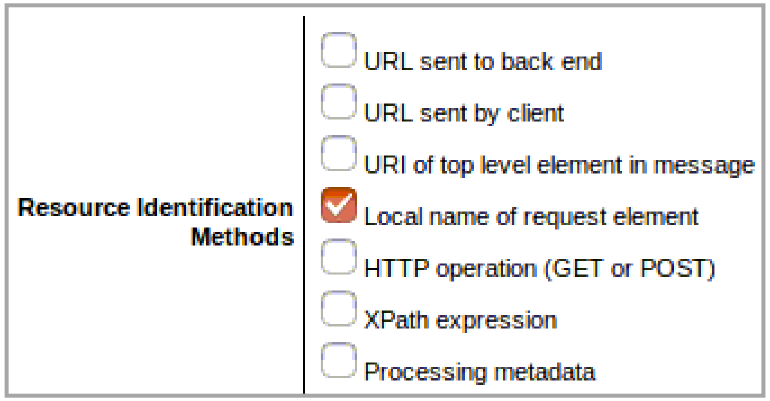
Method: None

Scenario 1: Authorize access to resources

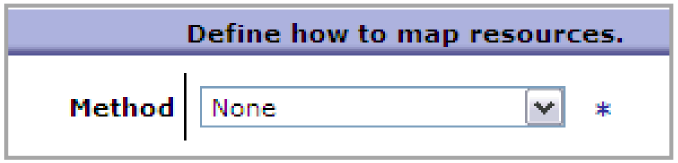
5. Select **Local name of request element** as the resource extraction method
   - The name of the child element in the SOAP body of the request is the request element name

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

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

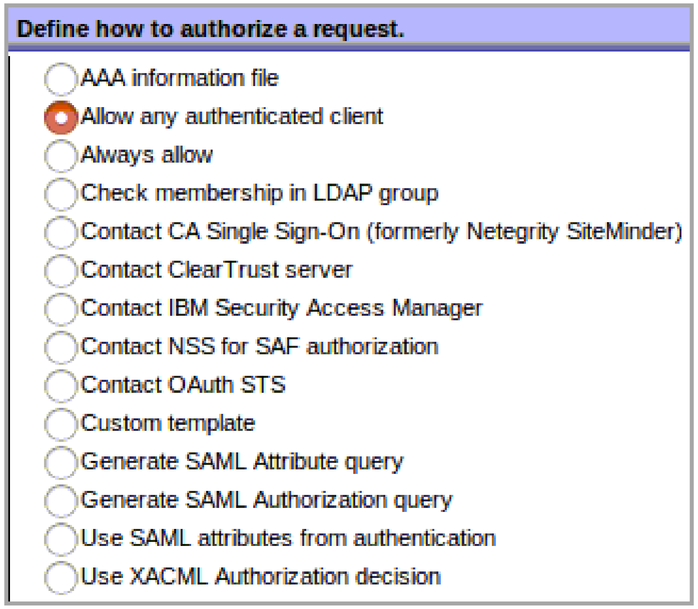


Resource Identification Methods	
	☐ URL sent to back end
	☐ URL sent by client
	☐ URI of top level element in message
	✓ Local name of request element
	☐ HTTP operation (GET or POST)
	☐ XPath expression
	☐ Processing metadata



Define how to map resources.

Method: None *



Define how to authorize a request.

- 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

Scenario 2: Security token conversion

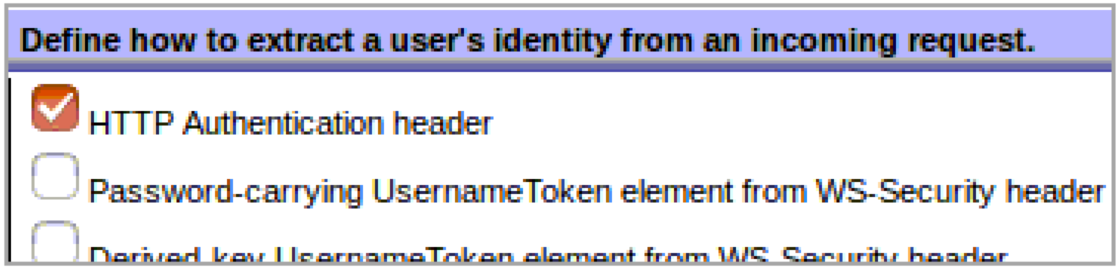
- Create an access control policy that handles client SOAP web service requests with the following conditions:
  - The client communicates to the DataPower gateway over a Secure Sockets Layer (SSL) connection
  - The HTTP BASIC-AUTH header information holds the identity of the requesting client
  - Generates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

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>
```

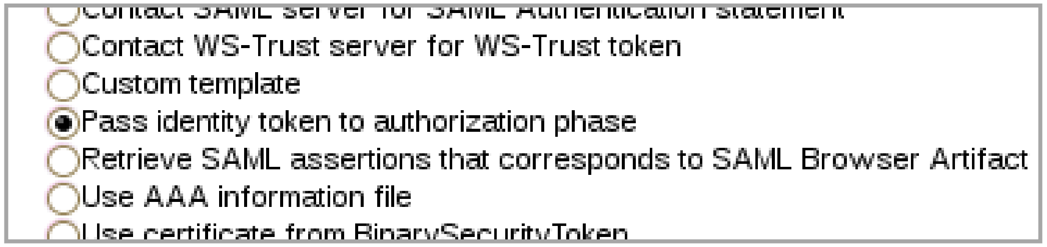
Scenario 2: Identify and authenticate the client

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**

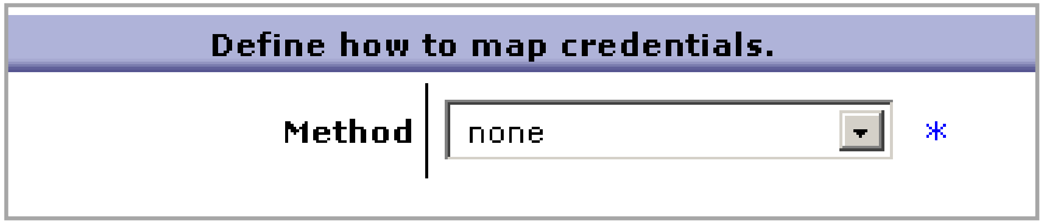


Define how to extract a user's identity from an incoming request.

- [✓] 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

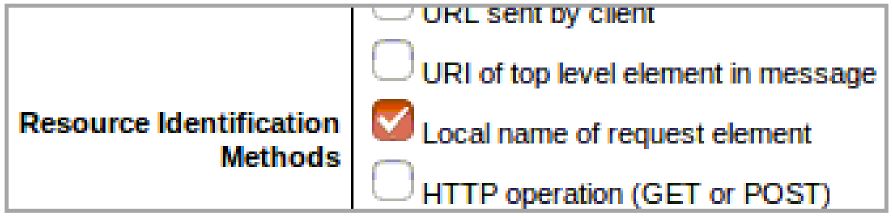


Define how to map credentials.

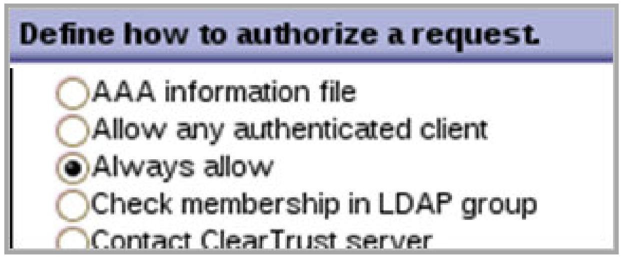
Method: none *

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

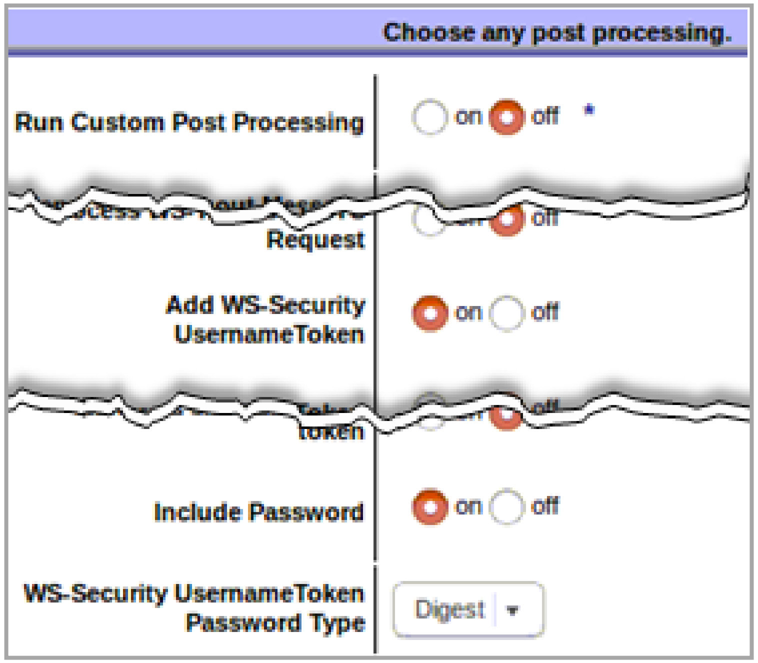


Resource Identification Methods	☐ URL sent by client ☐ URI of top level element in message ✓ Local name of request element ☐ HTTP operation (GET or POST)



Define how to authorize a request.

- ○ AAA information file
- ○ Allow any authenticated client
- ● Always allow
- ○ Check membership in LDAP group
- ○ Contact ClearTrust server



Choose any post processing.

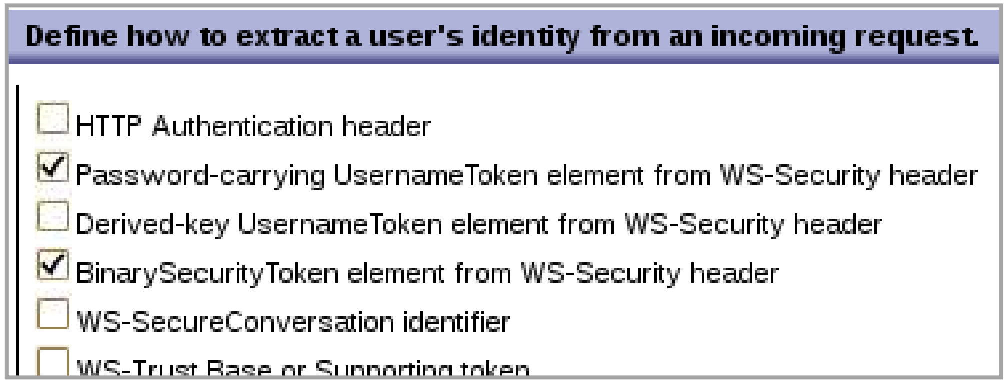
Run Custom Post Processing	on [✓ off] *
Request	[✓ off]
Add WS-Security UsernameToken	[✓ on] off
Token	[illegible]
Include Password	[✓ on] off
WS-Security UsernameToken Password Type	Digest

Scenario 3: Multiple identity extraction methods

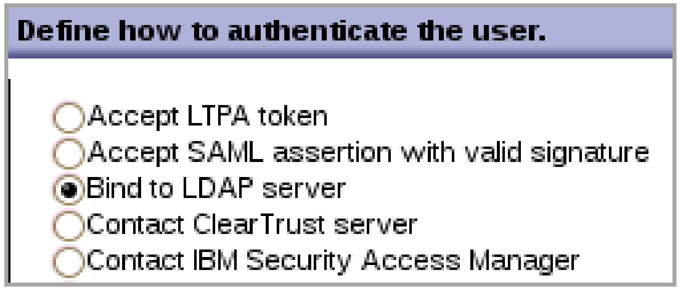
- Create an access control policy that handles client SOAP web service requests with the following conditions:
  - Uses either a WS-Security UsernameToken element or a BinarySecurityToken element from the WS-Security header to determine the client's identity
  - Verifies the identity of the client
 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

Scenario 3: Identify and authenticate the client

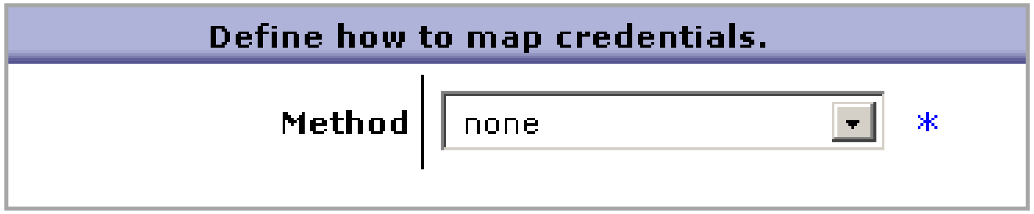
1. Create a AAA policy object on the DataPower gateway
2. Extract the client's identity from the Username element or a BinarySecurityToken
   - Separate WS-Security token profiles describe the structure of the UsernameToken and the BinarySecurityToken
3. For the authentication method, specify **Bind 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

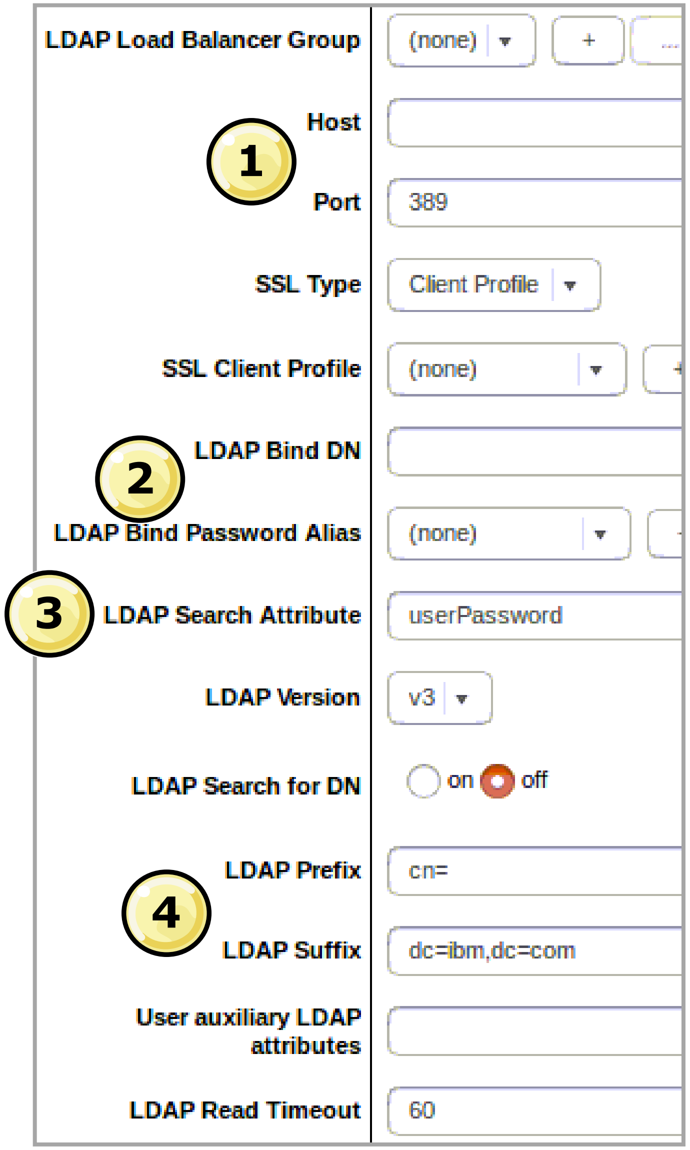

Define how to map credentials.

Method: none *

Scenario 3: LDAP details

When connecting to LDAP, further details are needed

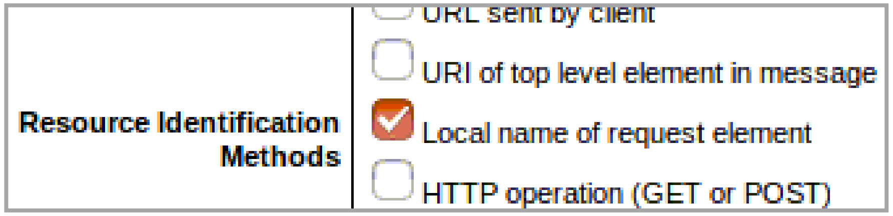
1. Specify the LDAP server URL and port
2. Indicate the **LDAP Bind DN** and **LDAP Bind Password Alias** for the LDAP query
3. Use the **LDAP Search Attribute** fields to verify the password digest from a WS-Security Username Token
4. Use the **LDAP Prefix** and **LDAP Suffix** fields to build the LDAP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



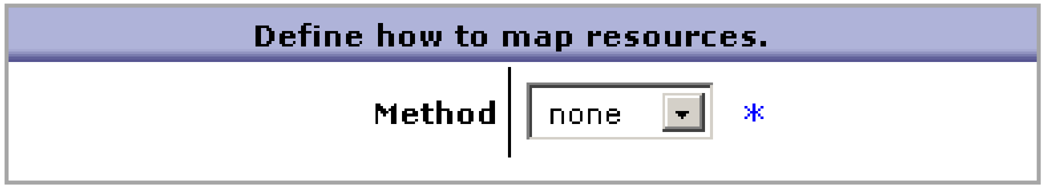
LDAP Load Balancer Group	(none)
Host	
Port	389
SSL Type	Client Profile
SSL Client Profile	(none)
LDAP Bind DN	
LDAP Bind Password Alias	(none)
LDAP Search Attribute	userPassword
LDAP Version	v3
LDAP Search for DN	on [✓ off]
LDAP Prefix	cn=
LDAP Suffix	dc=ibm,dc=com
User auxiliary LDAP attributes	
LDAP Read Timeout	60

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

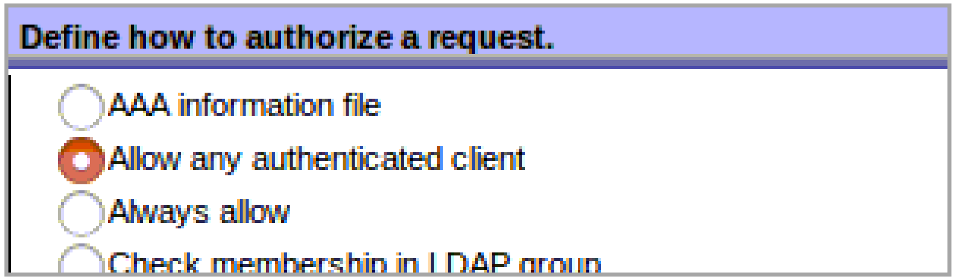


Resource Identification Methods
- [ ] URL sent by client
- [ ] URI of top level element in message
- [✓] Local name of request element
- [ ] HTTP operation (GET or POST)



Define how to map resources.

Method: none *

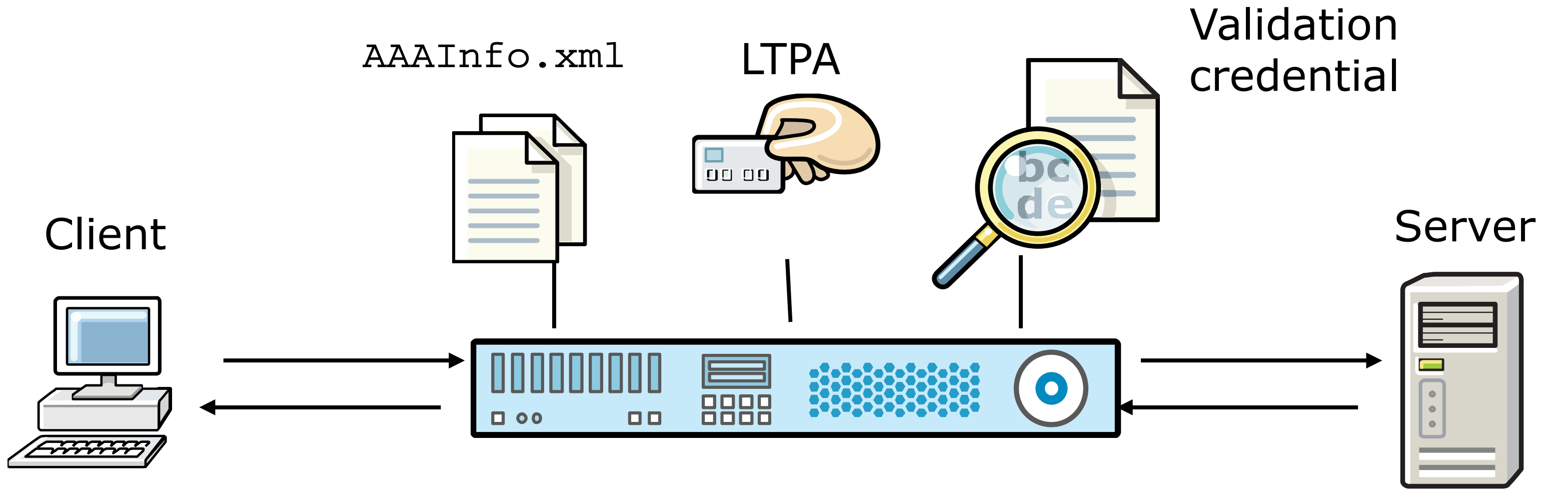


Define how to authorize a request.

- ○ AAA information file
- ● Allow any authenticated client
- ○ Always allow
- ○ Check membership in LDAP group

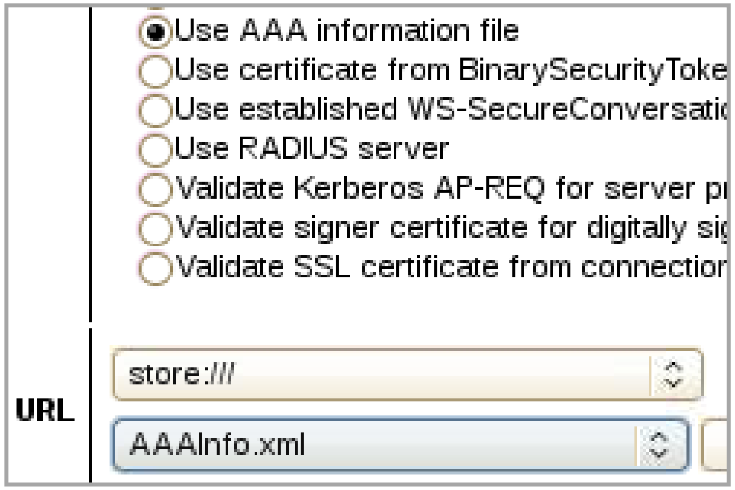
Internal access control resources

- 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



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



[✓ Use AAA information file]
Use certificate from BinarySecurityToke
Use established WS-SecureConversatio
Use RADIUS server
Validate Kerberos AP-REQ for server pr
Validate signer certificate for digitally sig
Validate SSL certificate from connectior

URL: store:/// AAAInfo.xml

Example AAA XML file

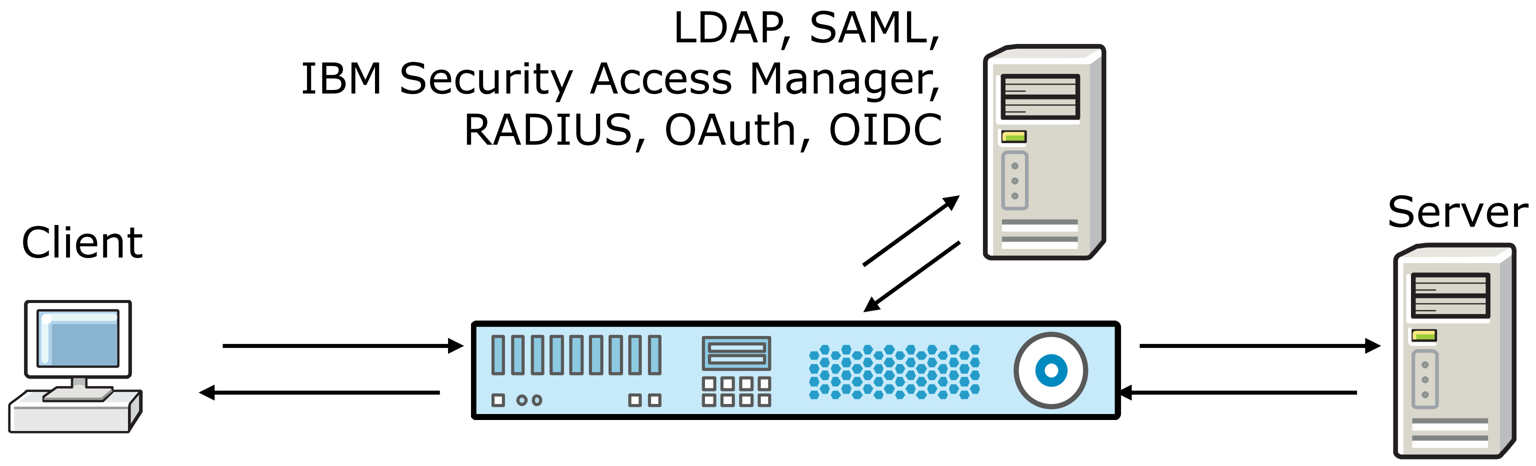
```
<aaa:AAAInfo   xmlns:aaa="http://www.datapower.com/AAAInfo">
       <aaa:FormatVersion>1</aaa:FormatVersion>
       <aaa:Filename>local:///AddressInfo.xml</aaa:Filename>
       <aaa:Summary>
               AAA file to validate credentials for Address users
       </aaa:Summary>

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

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

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

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)

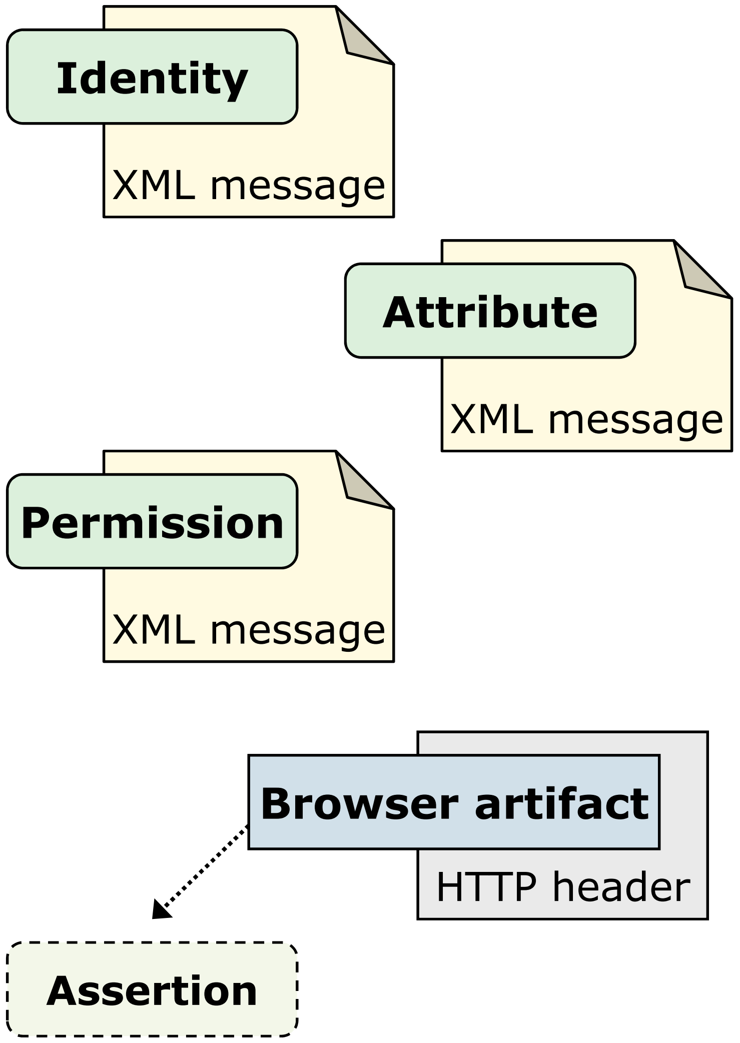
- A few facts about LDAP:
  - An LDAP directory is a tree of directory entries
  - The **distinguished name** (DN) is a unique identifier for entries
  - A **bind** operation authenticates the client by sending the client's distinguished name and password in cleartext
  - Use an SSL connection to keep 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

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



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: 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)*

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>
```

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)*

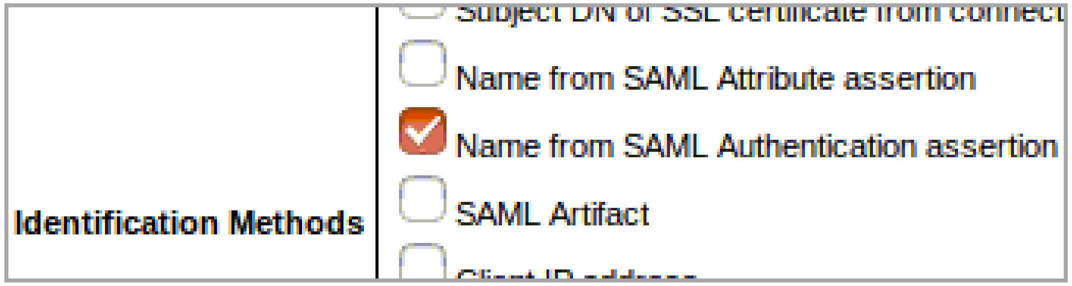
Scenario 4: SAML attribute statement (2 of 2)

. . . *(continued from previous slide)*

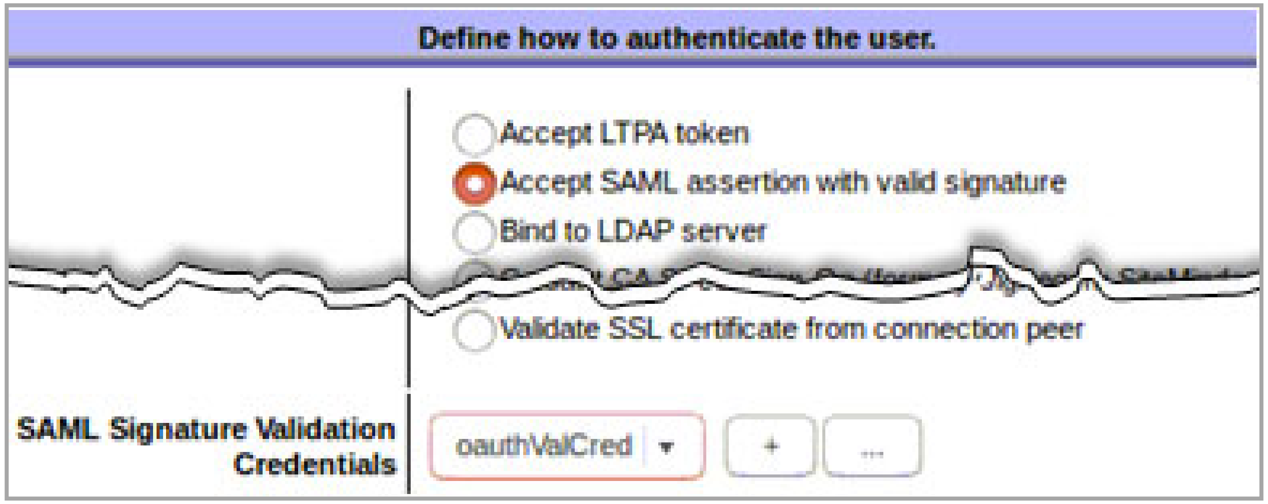
```
    <saml:Attribute
     AttributeName="EastAddressSearch"
     AttributeNamespace="http://address.training.ibm.com">
      <saml:AttributeValue>
        Query
      </saml:AttributeValue>
    </saml:Attribute>
  </saml:AttributeStatement>
</saml:Assertion>
```

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**



Identification Methods	Subject DN of SSL certificate from connect
	Name from SAML Attribute assertion
	[✓ Name from SAML Authentication assertion]
	SAML Artifact
	Client IP address



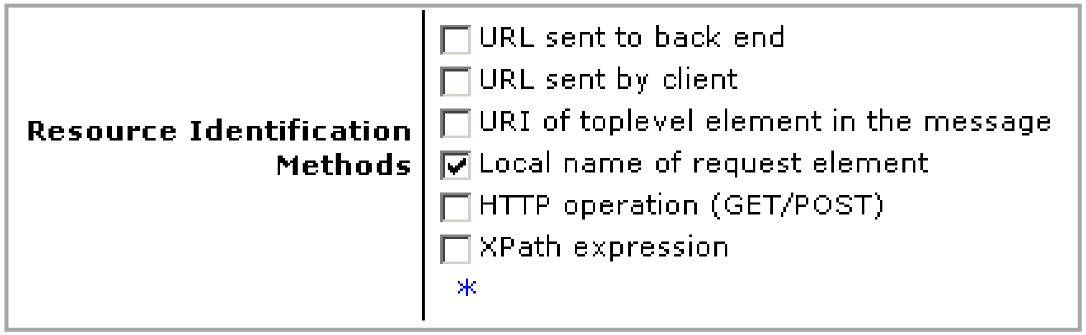
Define how to authenticate the user.

- Accept LTPA token
- [✓ Accept SAML assertion with valid signature]
- Bind to LDAP server
- Validate SSL certificate from connection peer

SAML Signature Validation Credentials: oauthValCred

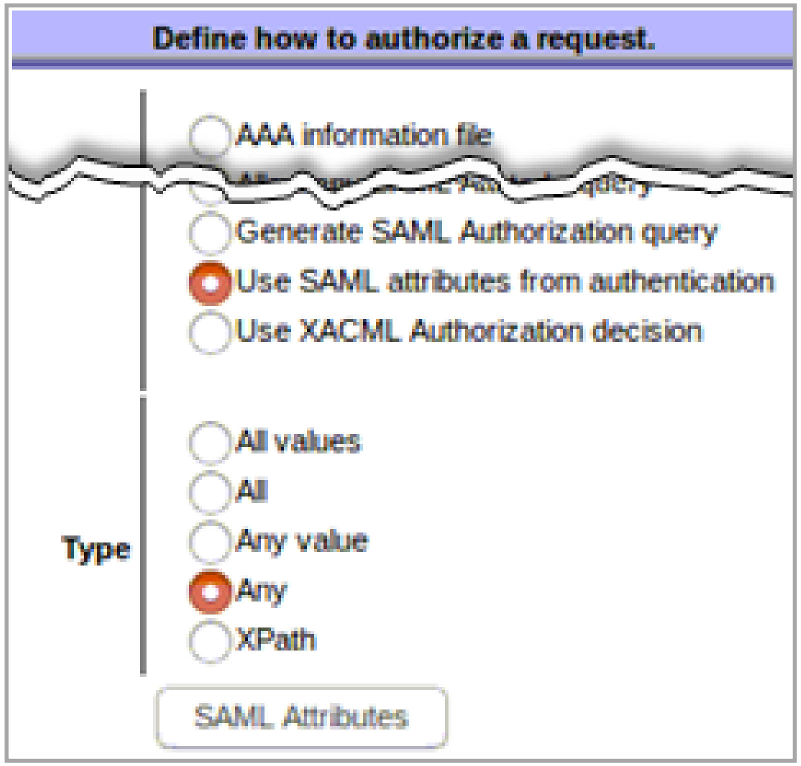
Scenario 4: Authorize access to resources

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
- [ ] 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

*



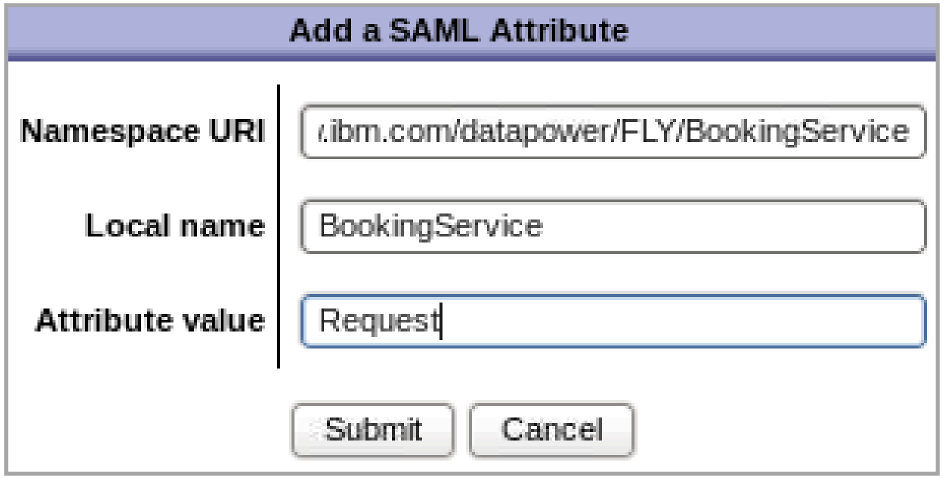
Define how to authorize a request.

- ○ AAA information file
- ○ Generate SAML Authorization query
- [● Use SAML attributes from authentication]
- ○ Use XACML Authorization decision

Type
- ○ All values
- ○ All
- ○ Any value
- [● Any]
- ○ XPath

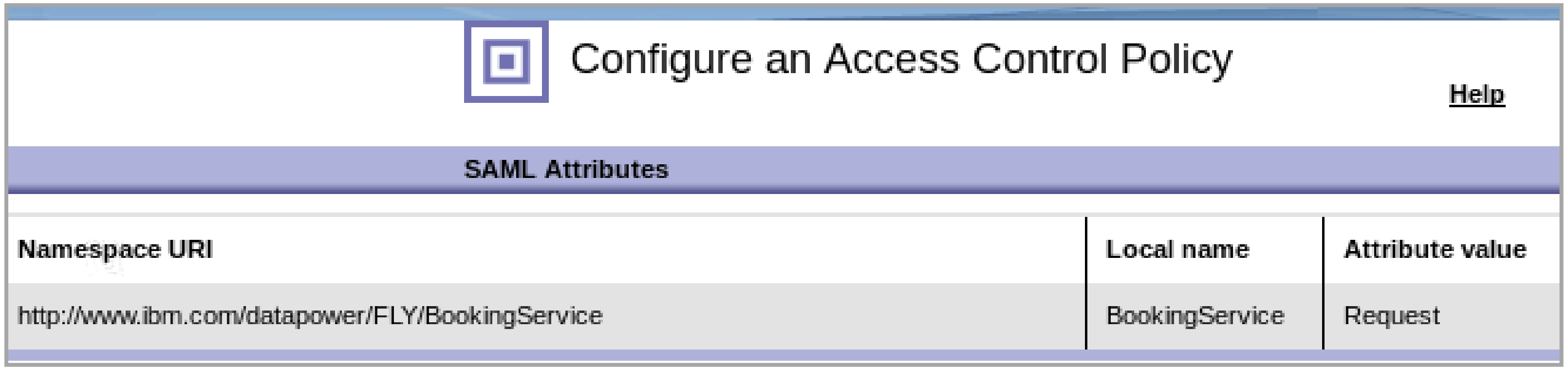
Scenario 4: Match SAML attributes

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	.ibm.com/datapower/FLY/BookingService
Local name	BookingService
Attribute value	Request



Configure an Access Control Policy

SAML Attributes

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

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

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.

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

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