WS-Policy security integration between DataPower and WebSphere Application Server

Russell Butek
John Rasmussen

November 18, 2009

This article shows you how to configure WebSphere DataPower SOA Appliance and WebSphere Application Server to implement WS-Policy for SOA service governance. User credentials are transformed in an LPTA common token format for authorization and single-sign-on between DataPower and an application hosted on WebSphere Application Server. Offloading policy management to DataPower lets WebSphere Application Server provide better application-level functionality, while DataPower provides enterprise-wide, high-performance service governance.

Introduction

Requests for enterprise resources have expanded to encompass not only trusted organizational clients, but distant and unfamiliar ones. The success of SOA has resulted in the publication of enterprise applications as services to be discovered and processed in a dynamic fashion.

SOA specifications such as Web Services Description Language (WSDL) have defined application method structure, while mechanisms such as IBM® WebSphere® Registry and Repository or Universal Description Discovery and Integration (UDDI) have been created to advertise the locations of services and the requirements to execute them. Services that were once confined to predefined and hard-coded application programming structures are now accessed in a loosely coupled fashion.

This exposure of resources has been accompanied by an effort to ensure robust authentication. The WS-Security specification has evolved to support the use of Security Assertion Markup Language (SAML), X.509 certificates, Kerberos tickets, and Username and other forms of identity transmission within a unified metadata standard that provides standardization for authentication and authorization.

As these tools have evolved, service providers have sought to establish organizational policies and governance that is application- and platform-agnostic. The WS-Policy protocol and its supporting specifications, such as Web Service Policy Framework, Policy Attachment, WS-SecurityPolicy and WS-ReliableMessagingPolicy, seek to establish a unifying methodology that may be discovered and consumed as services are discovered. In this way, access to groups of services can be controlled en masse rather than on an individual basis.
WS-Policy describes security and quality of service requirements in a machine and human readable format that facilitates the automated construction of service requests. Design-time decisions can be made to ensure that the proper credentials are passed to the service. Runtime decisions can be made for authorization of individual requests. Importantly, WS-Policy does not actually define the assertions' implementation, but rather defines the metadata that advertise the assertion, describing for example that a service requires a Username. It is up to the Policy Decision Point (PDP) implementation that interprets the WS-Policy metadata to make the actual assertion of the existence of the Username token (UNT).

This article will demonstrate implementation of WS-Policy for SOA service governance via the implementation and enforcement of a PDP configuration within the WebSphere DataPower® SOA Appliance (DataPower). DataPower will exchange identity information with an application hosted on WebSphere Application Server. By offloading policy management to DataPower, WebSphere Application Server is better able to provide application-level functionality, while DataPower provides enterprise-wide, high-performance service governance.

**Introduction to LTPA token**

Tokens are often used to convey identity within message traffic. Examples of tokens are SAML, X.509 certificate, and Kerberos tickets. Tokens can be managed and distributed by Secured Token Services (STS), as described by the WS-Trust specification.

Lightweight Third-Party Authentication (LTPA) tokens are used by IBM WebSphere and IBM Lotus Domino products. LTPA tokens contain an identity signed and symmetrically encrypted using a limited lifetime key that is shared with trusted entities. This procedure provides efficient confidentiality without the typical session key creation phase that establishes an ephemeral key for symmetric encryption used by asymmetric encryption technologies such as SSL/TLS. The shared key (shared secret) must be communicated between these trusted parties prior to use. The LTPA token is also used to provide single sign-on (SSO) within or across WebSphere Application Server cells.

LTPA tokens can be passed between processes and WebSphere Application Server applications via HTTP cookie headers, or within a WS-Security Binary Security Token. Its encrypted nature provides the confidentiality needed to protect the identity information within it.

There are several versions of the LTPA token. Version 1 tokens contain identity and realm information. The realm is used to associate the user registry used by WebSphere Application Server for authentication. Newer Version 2 tokens introduced in WebSphere Application Server V6 carry custom attributes, but custom programming is required to access and use it.

When a WebSphere Application Server or Domino application receives an LTPA token, it does not need to reauthenticate the user, but it may still need to access the user registry to create a complete Subject object containing information such as group associations.

The DataPower Appliance can decrypt and use LTPA identity, and create new LTPA tokens (The IBM Tivoli Access Manager WebSEAL product also provides this capability). The shared secret key must be managed between DataPower and WebSphere Application Server applications,
and issues such as the lifetime of a key must be addressed. WebSphere Application Server features, such as automatic key generation, may cause decryption failures if the new key is not communicated properly, and therefore it may be better to manage key generation via a scripted process or manual intervention. Because of their performance features and support across WebSphere Application Server and DataPower, LTPA tokens are often used when DataPower is positioned in front of WebSphere Application Server servers.

**Sample application**

DataPower fulfills a valuable role within a typical application environment. By using its purpose-built, hardware-optimized cryptographic functionality, DataPower can offload processor-intensive operations, such as digital signature integrity checking, and encrypting and decrypting messages for confidentiality. Offloading this intensive processing lets the application program stack more efficiently perform the message processing it is designed for.

Clients may make requests for services that require credential information in order to authenticate and authorize application access. DataPower is designed to authenticate a variety of identity tokens, including X.509 certificates, WS-Security Username tokens, SSL certificates, and Basic Authentication headers. Authorization can be performed against repositories such as LDAP directories or via applications such as IBM Tivoli Access Manager.

The sample application combines authentication with enforcement via WS-Policy policies. Figure 1 below shows a topology in which clients must submit requests containing WS-Security Username tokens over SSL, to ensure the confidentiality of identity information. Identities are authenticated and authorized for application access. All authenticated users are assigned a group identity, which is be stored in a signed LTPA token. This token is symmetrically encrypted using a shared secret that is known to DataPower and WebSphere Application Server. Symmetric encryption is orders of magnitude faster than asymmetric.

**Figure 1. Sample application architecture**

Once WebSphere Application Server receives the LTPA token, it decrypts using the shared secret, extracts the identity information contained within, and uses it to authenticate the request and
establish a JEE identity. This simple application merely echoes back a string contained within the request message and appends onto it the JEE identity provided by the runtime.

PDP implementations with DataPower and WebSphere Application Server are used to ensure compliance. Within DataPower, a policy validates the existence of a WS-Security Username token. Within WebSphere Application Server, a policy verifies the requirement of an LPTA token within the request message metadata. Our example begins with configuring WebSphere Application Server, followed by the DataPower implementation.

**Configuring WebSphere Application Server**

This section shows you how to configure a WebSphere Application Server V7 Web services provider to consume an LTPA token and set it as the container of the service requestor's identity.

Web service quality-of-service configuration in WebSphere Application Server V7 is done through policy sets and bindings. WebSphere Application Server policy terminology is slightly different from WS-Policy terminology. What WebSphere Application Server calls a policy, WS-Policy calls a policy expression -- a collection of metadata that describes a rule or requirement for a service. WebSphere Application Server also uses the term policy set, which is simply a collection of policy instances. WebSphere Application Server has a number of policies, such as WS-Security, WS-Addressing, and HTTP transport. Policy sets combine policies into single, configurable entities. For example, one of the prepackaged WebSphere Application Server policy sets, the Kerberos V5 HTTPS default policy set, consists of instances of WS-Security, WS-Addressing, and SSL transport policies.

A policy set tells you what the configuration is, but does not tell you how to achieve it. For example, it tells you that a SOAP request's body must be encrypted, but it does not tell you how to encrypt it -- it does not provide the certificate keystores. That's what the binding is for -- the binding is the entity that fills in the variable information such as keystores.

From this brief introduction, you can see that there are three entities in WebSphere Application Server that you will work with: a policy set, a binding, and the application. You attach a policy set to an application, and then assign a binding to that application. The rest of this section shows you how to create a policy set, create a binding, attach the new policy set to an application, and assign the binding to that same application.

**Create the LTPA policy set**

You could create a policy set from scratch, but WebSphere Application Server prepackages a number of policy sets, and one is nearly what you need: LTPA WSSecurity default. So instead of creating one, you can make a copy and modify that copy. The LTPA WSSecurity default policy set deals with an LTPA token, which is what you want. But it also signs and encrypts the message, which you do not need. Instead of signing the message, you will let SSL do the encryption. Figure 2 below shows the section of the Administrative Console from which you start the copy:

1. Navigate to Services => Policy sets => Application policy sets.
2. Select LTPA WSSecurity default and click Copy.
3. In the panel that follows, choose a name for your new policy set, such as LTPA over SSL.
4. Fill in a description of your choice, and then click OK:

   **Figure 2. Copy the prepackaged LTPA policy set**

   ![Image of Integrated Solutions Console](image)

   After you have created your copy, you will see it in the list of policy sets, as shown in Figure 3. The prepackaged policy sets are not editable, but your new LTPA over SSL policy set is editable. Click on LTPA over SSL to edit it. The window in Figure 4 opens.
Figure 3. Edit the copy

<table>
<thead>
<tr>
<th>Select</th>
<th>Name</th>
<th>Editable</th>
<th>Description</th>
</tr>
</thead>
</table>
|        | Kerberos V3 HTTPS default | Not editable | Policies: WSSecurity, SSLTransport, WSAddressing  
  - Message authentication: Using Kerberos V3 token  
  - Transport security: Using SSL for HTTP  
  - Follow the OASIS Kerberos Token Profile specification |
|        | LTPA WSSecurity default | Not editable | Policies: WSSecurity, WSAddressing  
  - Message integrity: Digitally sign body, timestamp, addressing headers, and LTPA token using RSA digital signing  
  - Message confidentiality: Encrypt body, signature, and LTPA token using RSA encryption  
  - Message authentication: Using LTPA token |
|        | LTPA over SSL          | Editable  | This policy is a copy of "LTPA WSSecurity default" with the following changes:  
  WS-security encryption and signing are removed; the "SSL transport" policy is added. |
|        | SSL WSTransaction      | Not editable | Policies: WSTransaction, SSLTransport  
  - Transactional integrity:  
    WS-Authorization and WS-BusinessActivity context propagation using SSL |

Figure 4 shows that your policy set is made up of two policy instances: WS-Addressing and WS-Security. You need to add SSL to the configuration:

1. Click the Add pull-down menu.
2. Select SSL transport from the list of available policies.

You can examine your new SSL policy instance by clicking on it. The default values are sufficient. That's all you have to do to enable SSL.
Figure 4. Edit the policy set

Now you must edit the WS-Security policy instance. From the window in Figure 4, navigate to WS-Security => Main policy:
Figure 5. Remove message protection

What we must do here is turn off message-level protection by unchecking that box. You are not doing WS-Security protection and we are not doing signing at all -- instead we are relying on SSL for encryption. That is all that we have to change in your version of the LTPA policy set.

Create the binding

WebSphere Application Server comes with prepackaged, default bindings. The default binding works as-is for the LTPA over SSL policy set: the LTPA token is validated and consumed by the SOAP engine, and the call to the Web service provider is successful. However, we want to go one step further. With the default binding, the identity is consumed by the SOAP engine -- it is not passed to the Web service implementation. We want the caller's identity -- the identity contained within the LTPA token -- to be passed to the implementation. So we need a bit more than is available in the default binding.

We could build a binding from scratch, but that would mean building a lot of the configuration that already exists within the default binding, so we will make a copy and edit that copy. To make a copy of the provider-side binding:

1. Navigate to Services => Policy sets => General provider policy set bindings.
2. Make a copy of Provider sample the same way that you copied the default LTPA policy set.
   Name the copy Demo provider sample.
3. At this point, as an optional step if you want to save a little time in the future, you can set the Demo provider sample to be the default provider binding. Navigate to Services => Policy sets => Default policy set bindings, and set Demo provider sample to be the default.

4. To make our change, step into the new binding's WS-Security policy instance. You will first see the left-hand panel:

**Figure 6. Create the caller**

5. The caller defines which token, if any, is used as the caller ID in the implementation. (A message may carry zero or more tokens -- in our case it will carry one.) Step to the next panel by clicking on Caller. By default, you can see that there is no caller identity. To add one, click New:
6. In this window, give the caller configuration any name you wish -- the name does not really matter. However, the values in the next two fields matter a great deal -- be careful of typos. These two fields define the fully qualified name of the token from the message to use as the caller identity, and these values must exactly match what will be in the SOAP message. Click Apply.

You might wonder where, exactly, you can find the proper values for the local part and namespace fields. For this article, we have provided them. But what are the values if you want to use other token types? If you thoroughly memorized the various WS-Security specs and the information center pages on these topics, you would simply know what they are. But most of us can't hold that much information in our heads, so there is a place in the admin console where the wizard fills in these values in for us. You can use that page as a reference:

1. Navigate to any binding that has a WS-Security policy.
2. Navigate from that binding to WS-Security => Authentication and protection => New token (under Authentication tokens) => Token generator.
3. Select the desired token type.
4. The wizard fills in the "Local part" and "Namespace URI" fields, as shown in Figure 8 below. Those fields correspond to the "Caller identity local part" and "Caller identity namespace URI" fields, respectively, on the panel in Figure 7 above.

**Figure 8. Find the fully qualified name of a given token type**

Attach the policy set

In the previous sections, you created a policy set and a binding. Now you must configure an application with that policy set and binding. For this example, use a variant of the EchoService sample, which is part of the JAX-WS samples that come with WebSphere Application Server. (If you chose not to install samples when you installed WebSphere Application Server, then you can use the version of this sample that comes with this article.)

1. Navigate to Services => Service providers => EchoService, as shown in Figure 9 below.
2. Select EchoService.
3. Click on Attach Policy Set and select your newly created policy set from the pull-down menu:
When you complete those steps, the LTPA over SSL policy set is shown to be the attached policy set, and the binding will be the default binding.

**Assign the binding**

If you set the Demo provider sample binding to be your default binding, then you don't have to do anything here; the default binding is automatically assigned. If you have not set it as the default, then:

1. Select the service on the window shown in Figure 9 above.
2. Click Assign Binding to pull down the menu of bindings.
3. Select Demo provider sample.

**The application used in this article**

As mentioned, the application used in this article is a variant of the EchoService application that's part of the samples that come with WebSphere Application Server. You are free to use the EchoService sample that comes with WebSphere Application Server, but with that version of the application, you have no ready proof that the identity in the LTPA token got to the service. For the purposes of demonstration, we modified the EchoService application so that it echoes not only the input string, but also the caller identity, as shown in the example result in Listing 6 in the DataPower section below, thus proving that the identity really does travel from DataPower to the
WebSphere Application Server application. If you would like to use the modified sample, you can download it at the bottom of the article.

**Integrating WebSphere Application Server and DataPower -- the LTPA key file**

**Export the LTPA key file from WebSphere Application Server**

Up to this point you have configured a WebSphere Application Server application to accept LTPA token request messages. The sender, in our case DataPower, must know something about the WebSphere Application Server LTPA configuration. This information is stored in an LTPA key file, so you must export the LTPA key file so that DataPower can later import it. To export this file:

1. Under Authentication, select Security => Global security => LTPA.
2. Under Cross-cell single sign-on as shown in Figure 10 below, enter a password of your choice. You must remember this password when you import the key file to DataPower.
3. Enter a key file name and click Export keys:

   ![Figure 10. Export the LTPA key file from WebSphere Application Server](image)

   There is one thing you must be aware of if you export LTPA keys. WebSphere Application Server automatically regenerates LTPA keys over time. When you export LTPA keys, at some point the keys in the exported file will no longer work. To keep them working, you can disable the automatic
generation of LTPA keys. When you do this, you must now manage the regeneration of keys on your own rather than relying on WebSphere Application Server to do it for you.

To disable automatic regeneration of keys, in Figure 10, click Key set groups under "Key generation" at the top of the panel. Then click on your key set group -- in this example NodeLTPAKeySetGroup. In the panel shown in Figure 11, clear the check box Automatically generate keys under "Key generation."

**Figure 11. Disable automatic LTPA key generation**

![Image of the Key set group management panel](image)

Import the LTPA key file into DataPower

Load the LTPA key file into the DataPower Appliance’s cert:// directory. You will see in the DataPower configuration discussion how the device's Authentication, Authorization, and Auditing (AAA) policy creates an LTPA token using this LTPA key.

**DataPower configuration**

DataPower is designed to act as a policy enforcement point (PEP) for WS-Policy, which is supported within the Web Service Proxy (WSP) service object. In DataPower firmware release 3.7.3 or later, the following WS-Policy specifications are supported:

- WS-Policy 1.2 and 1.5
- Web Service Policy Framework and Policy Attachment 1.5
- WS-SecurityPolicy 1.2 and 1.5
- WS-ReliableMessagingPolicy 1.2
WS-Policy Attachment provides support for WS-Policy authored within the WSDL document. In addition, policies may be attached to various WSDL subjects such as message, service, binding, or operation subjects using the WS-Proxy GUI. Support for standard WS-Policy domains such as WS-Security is implemented via templates supplied in the DataPower Appliance’s store://policies//templates directory. There you will find predefined templates to perform standard policy actions, such as enforcing digital signatures, encryption, presence of Username token, and using secured protocols among other policy templates.

In the DataPower configuration below, you will configure a WSP to support the requirements of the sample application and its interaction with the WS-Policy restrictions imposed by the WebSphere Application Server application. These are:

- Use WS-Policy to require Username token on client request
- Require secured transport on client request
- Convert Username token to WS-Security header with LTPA token

Basic configuration of the WSP is assumed, so the emphasis here is on the primary WS-Policy requirements, and some basic unrelated steps are not shown. For more information about basic WSP configuration, see WebSphere DataPower SOA Appliances documentation, or the WebSphere DataPower SOA Appliance Handbook.

Incoming request message configuration

The first step configuring the WSP is creating the WSP and assigning the WSDL. The service on WebSphere Application Server exposes the WSDL using the URL https://9.30.197.37:9443/WSSampleSei/EchoService?WSDL. The WSDL is obtained and loaded onto the device’s local:// directory via the File Management WebGUI. Figure 12 shows the assignment of the WSDL to the newly created WSP UNT-LTPA-Proxy:

Figure 12. WSDL Assignment to WSP
DataPower supports alternative methods of WSDL assignment. You can use Universal Description Discovery and Integration (UDDI) or WebSphere Service Registry and Repository to fetch the WSDL. WebSphere Service Registry and Repository can be used to establish polling intervals to fetch changes to WSDL. For example, if the location of the remote service (the WebSphere Application Server application) changes, the WSP could be configured to automatically use the new URL. For more information about UDDI or WebSphere Service Registry and Repository, see WebSphere DataPower SOA Appliances documentation.

The WSDL contains a single service that exposes a single port for message traffic. This port is exposed over HTTPS and is the endpoint where DataPower will forward validated requests to the EchoService. Listing 1 shows the service section from the WSDL:

Listing 1. WSDL service port assignments

```xml
<wSDL:service name="EchoService">
  <wSDL:port name="EchoServicePort" binding="tns:EchoSOAP">
  </wSDL:port>
</wSDL:service>
```

DataPower can expose multiple front-side protocol ports for receiving client traffic. In this case an HTTPS port is provided. By default, the remote address contains the endpoint information from the WSDL port. This can be changed if required, and can also be dynamically assigned if, for example, the endpoint is different for different classes of request, such as a high-speed service for "gold" clients, and a lower quality service provider for normal traffic. Dynamic assignment is performed via the DataPower policy Route action or within XSLT in a Transformation action. Figure 13 shows the assignment of local and remote endpoints:

Figure 13. Local and remote endpoint assignments

The local endpoint handler is a Front Side Protocol Handler (FSH) named untLTPA_HTTPS_FSH -- a basic HTTPS FSH that accepts traffic over port 8082. You have now defined a simple WSP that accepts HTTPS traffic and forwards it to HTTPS://9.30.197.37:9443/WSSSampleSei/EchoService.
The next configuration step is to use WS-Policy to ensure that the client requests contain the Username token as required by our sample application. All WS-Policy assignments are made from the WSP Policy tab. Policy can be attached at each level of the WSDL: service, proxy, operation, and as the WSDL itself. When attached, it will be enforced at all subsequent levels -- for example, policy attached at the service is operation level. You need to ensure that all traffic for all services contains the UNT, which you do via the WS-Policy button:

**Figure 14. WSP proxy tab**

The requirements are to ensure that the UNT is provided on all requests from the client. As mentioned, DataPower supports WS-SecurityPolicy 1.2. The [WS-SecurityPolicy 1.2 specification](https://www.w3.org/2002/ws/policy) contains Username Token assertion policies that may be used to assert the existence of this token. DataPower being a PDP (policy decision point) provides the implementation for this and other token assertion policies.

DataPower contains a sample template in the store://policies//templates directory named wsp-sp-1-1-usernametoken.xml that is close to our requirement.

The WS-SecurityPolicy 1.2 specification contains options for many variations of the UsernameToken assertion. For example, variations of the sp:IncludeToken attribute changes the requirement of the existence of the UNT on request and/or response messages. Not every variation is supported by a pre-existing template on DataPower, so in some cases you may need to modify the DataPower templates. There is a template on DataPower to assert the existence of the UNT on request and response messages. There is no preexisting template for assertion on UNT on the request message only, but you can create one.

Listing 2 shows a section of this policy. This policy is looking for a UNT in the 1.1 format. The template also contains a policy that looks for the 1.0 format UNT.
Listing 2. wsp-sp-1-1-usernametoken.xml snippet

```xml
<wsp:All>
  <sp:SupportingTokens>
    <sp:UsernameToken sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512/IncludeToken/Always">
      <wsp:Policy>
        <sp:WssUsernameToken11/>
      </wsp:Policy>
    </sp:UsernameToken>
  </sp:SupportingTokens>
</wsp:All>
```

In the WS-SecurityPolicy 1.2 specification in section 5.1.1 Token Inclusion Values is the following description (Table 1) of token expectations as determined by the sp:IncludeToken attribute:

**WS-Security Policy sp:IncludeToken description**

**http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never**

The token MUST NOT be included in any messages sent between the initiator and the recipient; rather, an external reference to the token should be used.

**http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Once**

The token MUST be included in only one message sent from the initiator to the recipient. References to the token MAY use an internal reference mechanism. Subsequent related messages sent between the recipient and the initiator may refer to the token using an external reference mechanism.

**http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient**

The token MUST be included in all messages sent from initiator to the recipient. The token MUST NOT be included in messages sent from the recipient to the initiator.

**http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToInitiator**

The token MUST be included in all messages sent from the recipient to the initiator. The token MUST NOT be included in messages sent from the initiator to the recipient.

**http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Always**

The token MUST be included in all messages sent between the initiator and the recipient. This is the default behavior.

The value of http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200512/IncludeToken/Always when assigned to sp:IncludeToken attribute requires a UNT on both the request and response to the client. The requirement is to require only the UNT on the request. Therefore you can change the sp:IncludeToken attribute to http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient, which only requires the UNT on the request message and not the response.

Rather than modify the policy in store://policies//templates (which you don't have permission to do), copy this to local://, rename to wsp-sp-1-1-usernametoken_AlwaysToRecipient.xml along with making the aforementioned AlwaysToRecipient change.

Now that you've created a customized policy, it needs to be attached to the service level of the WSDL. Figure 15 shows the pop-up window that results from clicking the WS-Policy button. The
Sources tab is used to assign the policy. Navigating to the local:// directory lets you select the newly created wsp-sp-1-1-userrnametoken_AlwaysToRecipient.xml policy.

Some policy documents will have multiple policies within differentiated by a unique wsu:Id attribute. When encountering such a document, designate which to use by selecting the appropriate wsu:Id. In this example there is only one. Click Attach Source and Apply for the WSP to complete the assignment:

**Figure 15. Attaching WS-Policy to WSDL**

![Figure 15. Attaching WS-Policy to WSDL](image)

The UNT policy does not require any additional information, but some policies require parameters for completion. For example, a policy that enforces a digital signature requires the name of a DataPower Crypto Certificate to verify the signature. The Processing tab allows the assignment of these properties. Additionally, the Enabled Subject tab lets you fine tune the WSDL elements and message phases to which the policy is enforced. For example, you may wish to only perform policy enforcement on the request phase of a message, and not on the response.

If you submit a simple request to this service via DataPower without the UNT, a violation occurs. Listing 3 shows an EchoRequest without the UNT:
Listing 3. Sample EchoRequest without UNT

```xml
<?xml version="1.0" encoding="UTF-8"?>
<S11:Envelope
 xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
 xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
 xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:tns="http://com/ibm/was/wssample/sei/echo/
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <S11:Body>
 <tns:echoStringInput>
 <echoInput> Are you talkin' to me?</echoInput>
 </tns:echoStringInput>
 </S11:Body>
</S11:Envelope>
```

Using cURL ([http://www.haxx.se/](http://www.haxx.se/)), you can submit this request and see the results. Listing 4 shows the response from DataPower:

Listing 4. Error message return without UNT

```bash
curl -k -d @echoRequestNoUNT.xml
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Body><env:Fault><faultcode>env:Client</faultcode><faultstring>
 Required elements filter setting reject: expression */[local-name()='Envelope'
 and (namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/'
 or namespace-uri()='http://www.w3.org/2003/05/soap-envelope')]/*[local-name()=
 'UsernameToken' and namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd']
 was not satisfied (from client)</faultstring></env:Fault></env:Body></env:Envelope>
```

Adding a UNT to the request as seen in Listing 5 should fulfill the requirements of the newly added policy. The UNT has been added to the wsse:Header, wsse:Security element:

Listing 5. EchoRequest with UNT

```xml
<?xml version="1.0" encoding="UTF-8"?>
<S11:Envelope xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
 xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
 xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:tns="http://com/ibm/was/wssample/sei/echo/
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <S11:Header>
 <wsse:Security>
 <wsse:UsernameToken wsu:Id="username">
 <wsse:Username>fred</wsse:Username>
 <wsse:Password>flintstone</wsse:Password>
 </wsse:UsernameToken>
 </wsse:Security>
 </S11:Header>
 <S11:Body>
 <tns:echoStringInput>
 <echoInput> Are you talkin' to me?</echoInput>
 </tns:echoStringInput>
</S11:Envelope>
```
Outgoing request message configuration

Before sending a request to the WebSphere Application Server application, there is a little more work to do. WebSphere Application Server is enforcing the identity of the requestor in the LTPA token. In addition, while you may let several clients make requests, WebSphere Application Server is only configured for a single user for this application, so you must map validated requests to a WebSphere Application Server accepted distinguished name via the AAA action of the processing policy. Figure 16 shows the default request rule with the addition of an AAA action:

Figure 16. Proxy policy with AAA action

The AAA policy accepts the client's credentials from the UNT as shown in Figure 17, which demonstrates the identity extraction phase of the AAA action:

Figure 17. AAA action identity extraction
While a real-world application would use a mechanism such as LDAP to authenticate the UNT credentials, this example uses an AAA Info XML file stored on the device. Figure 18 shows the authentication phase:

**Figure 18. AAA action authentication**

The AAA Info file provides a convenient way to assign the new credential for WebSphere Application Server. You can use alternative methods, such as Tivoli Federated Identity Manager, Secured Conversation, or various custom methods, such as XPath inquiry into the request document or customer XSLT. Figure 19 shows the mapping from the AAA Info file:

**Figure 19. AAA info credential mapping**

The final step in AAA processing is converting the UNT to the LTPA token. In this post-processing phase, Generate LTPA Token has been selected. The default option is to store the LTPA Token in an HTTP Cookie header. Optionally, the token can be stored in the WS-Security header element, and that option has been chosen here. DataPower provides a dropdown box (LTPA Token Version), which is used to designate the LTPA token format. The goals are to select the token version (V1/V2), and whether to use an HTTP Cookie or a Binary Security Token (BST) to contain the token. In addition there are two forms of BST (typically used with Web service traffic) available with different namespace declarations.

The LTPA Token Version options of Domino, WebSphere Version 1 and WebSphere Version 2 are self evident. WebSphere Version 1 FIPS provides enhanced security compliant with the Federal
Information Processing Standard (FIPS) for the Version 1 token (the Version 2 token is inherently FIPS compliant). And WebSphere V7 Version 2 is used to create a BST with the wws/LTPAv2 namespace. These values where undergoing review when this article was published, so you should check the DataPower product guides for the latest information on your firmware revision. Figure 20 shows the LTPA selection:

**Figure 20. AAA post-processing for LTPA token creation**

Referring back to the LTPA introduction, there are multiple versions of the LTPA Token. We've selected the WebSphere V7.0 Version 2 format. In addition, the LTPA Key File as obtained from WebSphere Application Server (see discussion under WebSphere Application Server configuration) is loaded onto the device’s cert:// directory and the password is entered. LTPA User Attributes could have been entered. These name-value pairs require application programming in the WebSphere Application Server application for consumption and interpretation.

Completing the AAA action finalizes the configuration of the WSP policy. Requests submitted to the application through the DataPower appliance are now validated by the WSP and the WS-Policy. Username tokens are converted to LTPA Tokens by the AAA policy and submitted to the WebSphere Application Server application. Listing 6 shows an example of a request and response using cURL. The application responses with the echoed string and the username (wsadmin) as extracted from the LTPA Token.
Listing 6. Submission of request through DataPower

curl -k -d @echoRequest.xml https://9.33.96.224:8082/WSSampleSei/EchoService -H "Content-type: text/xml" -H "SOAPAction: echoOperation" --key WSTC-privkey.pem --cert WSTC-sscert.pem

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:echoStringResponse xmlns:ns2="http://com/ibm/was/wssample/sei/echo/">
      <echoResponse>JAX-WS==>> Are you talkin' to me? (user: wsadmin)</echoResponse>
    </ns2:echoStringResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Conclusion

WS-Policy has been designed to provide enterprise-wide SOA governance. This article demonstrated its use and implementation within the WebSphere DataPower SOA Appliance for the enforcement of Username Tokens and within WebSphere Application Server for the enforcement of LTPA tokens. LTPA tokens are an effective and efficient method of single sign-on, by providing identity information that does not need to be re-authenticated.
Related topics

- IBM Redbook: Web Services Feature Pack for WebSphere Application Server V6.1
- Web Services Policy 1.5 -- Primer
- Web Services Policy 1.5 -- Framework
- Web Services Policy 1.5 -- Attachment
- WS-Security Policy 1.2
- WebSphere DataPower SOA Appliances product page
- WebSphere DataPower SOA Appliances product library
- WebSphere DataPower SOA Appliances documentation
- IBM Support
- WebSphere DataPower SOA Appliance Handbook
- IBM Redbook: IBM WebSphere DataPower SOA Appliances, Part I: Overview and getting started
- IBM Redbook: IBM WebSphere DataPower SOA Appliances Part II: Authentication and Authorization
- IBM Redbook: IBM WebSphere DataPower SOA Appliances Part IV: Management and Governance
- IBM Redpaper: WebSphere DataPower SOA Appliances: The XML Management Interface
- WebSphere Application Server product page
- WebSphere Application Server information center
- WebSphere Application Server documentation library
- WebSphere Application Server support
- developerWorks on Twitter
- Most popular WebSphere trial downloads
- WebSphere on-demand demos
- WebSphere-related books from IBM Press

© Copyright IBM Corporation 2009
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)