Implementing message flow security in WebSphere Message Broker V7

Shenfu (Mike) Fan

February 24, 2010

You can configure a broker in WebSphere Message Broker V6.1 or later to perform access control for individual messages in a message flow by using an external security provider. This article describes security at the message flow level and shows you how to implement message flow security.

Message flow security overview

In IBM® WebSphere® Message Broker V6.1 or later, (hereafter called Message Broker) a security manager enables a broker to perform end-to-end processing of an identity carried in a message through a message flow. With this access control based on the identity associated with a message, security is independent of both transport type and message format. Without enablement of this message flow security, the default security facilities in WebSphere Message Broker are based on the security provided by the transport mechanism. A broker processes all inbound messages in a message flow by default using the broker service user ID as a proxy identity, and any identity carried in the messages is ignored. The security manager performs the following functions:

- Extract the identity from an inbound message
- Authenticate the identity
- Map the identity to an alternative
- Authorize either the alternative or the original identity to access the message flow
- Propagate either the alternative or the original identity with an outbound message

To perform the authentication, mapping, and authorization, an external security provider is required. Message Broker supports two external security providers: Lightweight Directory Access Protocol (LDAP) for authentication and authorization, and IBM Tivoli® Federated Identity Manager (TFIM) for authentication, mapping, and authorization.

The security functions are defined using security profiles, which are normally created by a broker administrator. A security profile specifies whether authentication, mapping, authorization, and propagation are performed on the identity of the message in a message flow, and if so, which external security provider is used. Multiple security profiles can be created on a broker and associated in a message flow. These security profiles are accessed by the security manager at run time.

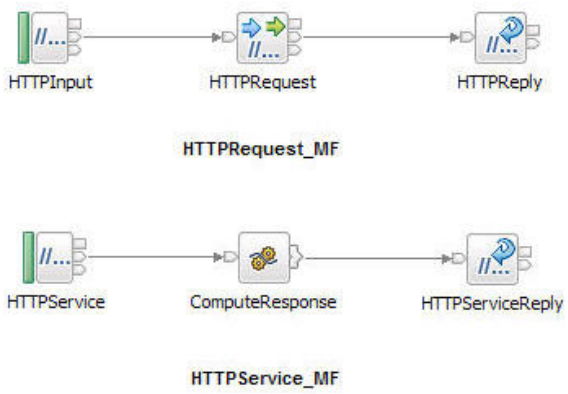
The security functions defined in a security profile are performed on input and output/request nodes of a message flow. The input nodes that support runtime identity extraction are MQInput, HTTPInput, and SOAPInput. The output/request nodes that support runtime identity propagation are MQOutput, HTTPRequest, SOAPRequest, and SOAPAsyncRequest. Each of the input and output/request nodes has a security profile property that you can set using the Broker Archive editor in the WebSphere Message Broker Toolkit before deployment.

In the following sections, an example shows you how to set up message flow security. To benefit from this article, you should have basic knowledge of Message Broker, and know how to create, deploy, and debug message flows.

Message flow creation

Two pre-built message flows are provided, as shown in Figure 1. One accepts HTTP requests and another provides services. These message flows are used to demonstrate the security configuration. In the HTTPRequest_MF message flow, the HTTPInput node receives an XML message at http://localhost:7080/CustomerOrders/ProcessOrderServices and the HTTPRequest node invokes a service at http://localhost:7080/OrderRequestService, which is provided by the HTTPService_MF message flow. In the HTTPService_MF message flow, the ComputeResponse node generates a response XML message, which is sent back to HTTPRequest_MF by the HTTPServiceReply node. In turn, the HTTPReply node in HTTPRequest_MF delivers the response to the caller. The message flows are available in the Project InterChange file MsgFlowSecurity_PI.zip, which you can download below. For information on creating a message flow, see Creating a message flow in the Message Broker information center.

Figure 1. Pre-built message flows



Message flow security configuration

To set up message flow security, you need to complete the following tasks:

1. Add user and group entities in an external security provider
2. Create security profiles
3. Configure identity extraction on message flows
4. Associate the security profiles with the message flows

1. Add user and group entities in an external security provider

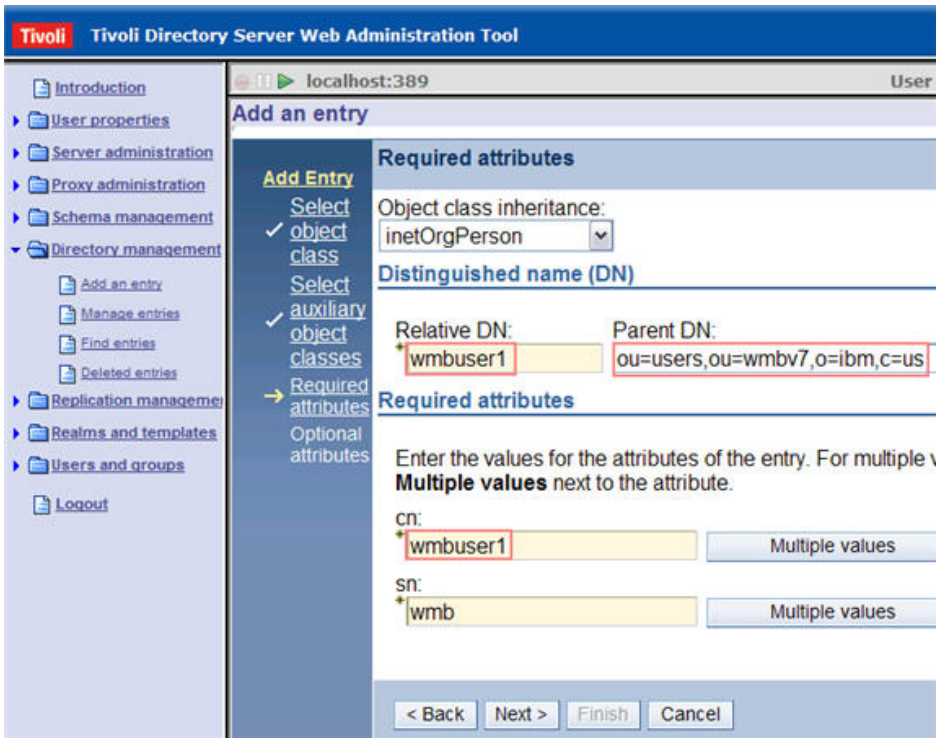
As mentioned above, the message flow security implementation depends on an external security provider. The example in this article uses Tivoli Directory Server V6.2 (LDAP) as the external security provider. This LDAP server is running on localhost (Windows® XP) at the default port 389. Installation and configuration of Tivoli Directory Server is not covered in this article. You need to create the entities listed in Table 1 in LDAP. These entities are under the directory tree structure ou=users,ou=wmbv7,o=ibm,c=us that has been built.

Table 1. Entities in LDAP

User	uid	Distinguished Name (DN)	Password	Group
wmbuser1	wmbuser1	cn=wmbuser1,ou=users,ou=wmbv7,o=ibm,c=us	wmbuser1	cn=authorized,ou=users,ou=wmbv7,o=
wmbuser2	wmbuser2	cn=wmbuser2,ou=users,ou=wmbv7,o=ibm,c=us	wmbuser2	
wmbuser3	wmbuser3	cn=wmbuser3,ou=users,ou=wmbv7,o=ibm,c=us	wmbuser3	cn=authorized,ou=users,ou=wmbv7,o=
wmbuser4	wmbuser4	cn=wmbuser4,ou=users,ou=wmbv7,o=ibm,c=us	wmbuser4	
wmbuser5	wmbuser5	cn=wmbuser5,ou=users,ou=wmbv7,o=ibm,c=us	wmbuser5	cn=authorized,ou=users,ou=wmbv7,o=

1. Log in to the Tivoli Directory Server Web Administration Tool. Expand **Directory management**, click **Add an entry** to create the user wmbuser1, and then select the object class **inetOrgPerson**.
2. On the Required attributes page, fill out all the fields as shown in Figure 2:

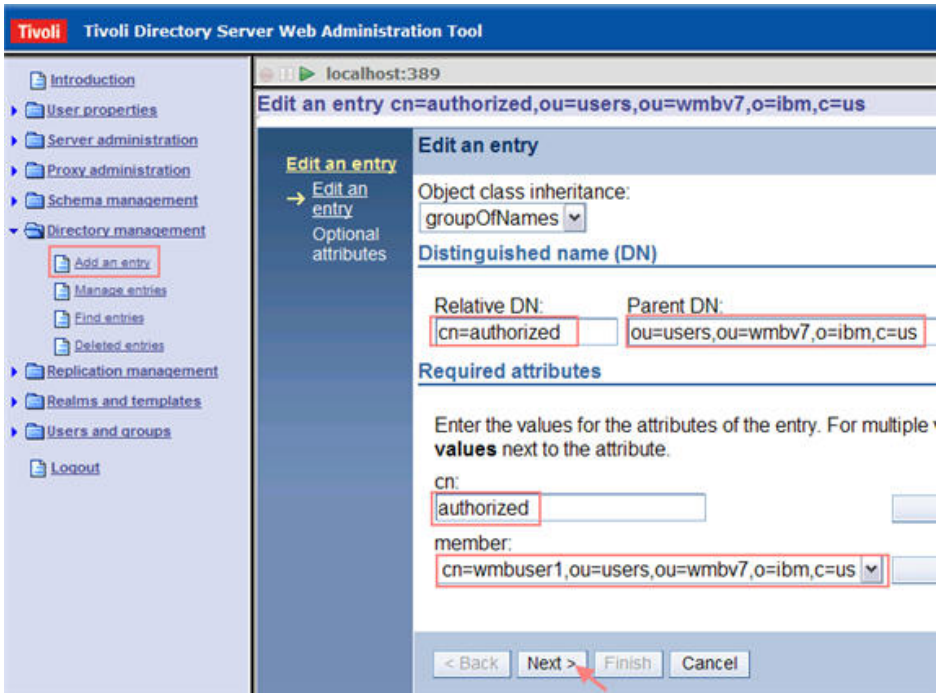
   Figure 2. Create a user in LDAP



3. On the Optional attributes page, fill out at least the uid and userPassword fields, then click **Finish**.

4. Repeat the same steps to add the rest of users.
5. To create the authorized group, click **Add an entry** and select the object class **groupOfNames**.
6. On the Required attributes page, fill out the fields as shown in Figure 3 below. Only wmbuser1, wmbuser3, and wmbuser5 are added as the group members. Click **Finish**:

Figure 3. Create a group in LDAP



2. Create security profiles

Create a security profile using either an editor in Message Broker Explorer, or the command mqsicreateconfigurableservice on the Message Broker Command Console. Create two security profiles LDAP_SP1 and LDAP_SP2 as shown in Table 2 -- one using the Explorer and the other using the command. Both of the security profiles are associated with LDAP. The only difference between them is the user password -- plain text in LDAP_SP1, masked with **** in LDAP_SP2.

Table 2. Details of security profiles

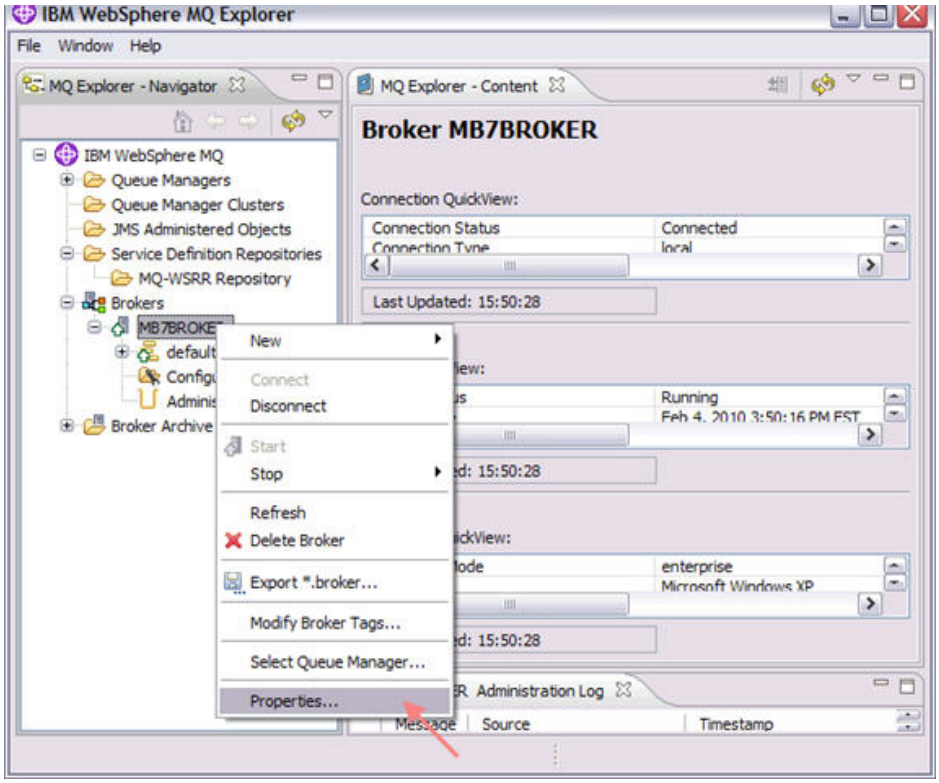
SecurityProfile	AuthenticationAuthenticat Config	Mapping	AuthorizationAuthorizati Config	Propagation	PasswordValue
LDAP_SP1	LDAPldap://localhost:389/ou=users,ou=wmbv7,o=ibm,c=us	NONE	LDAPldap://localhost:389/cn=authorized,ou=users,ou=wmbv7,o=ibm,c=us	TRUE	PLAIN
LDAP_SP2	LDAPldap://localhost:389/ou=users,ou=wmbv7,o=ibm,c=us	NONE	LDAPldap://localhost:389/cn=authorized,ou=users,ou=wmbv7,o=ibm,c=us	TRUE	MASK

You cannot perform the identity mapping using LDAP -- you must use Tivoli Federated Identity Manager.

Using Message Broker Explorer

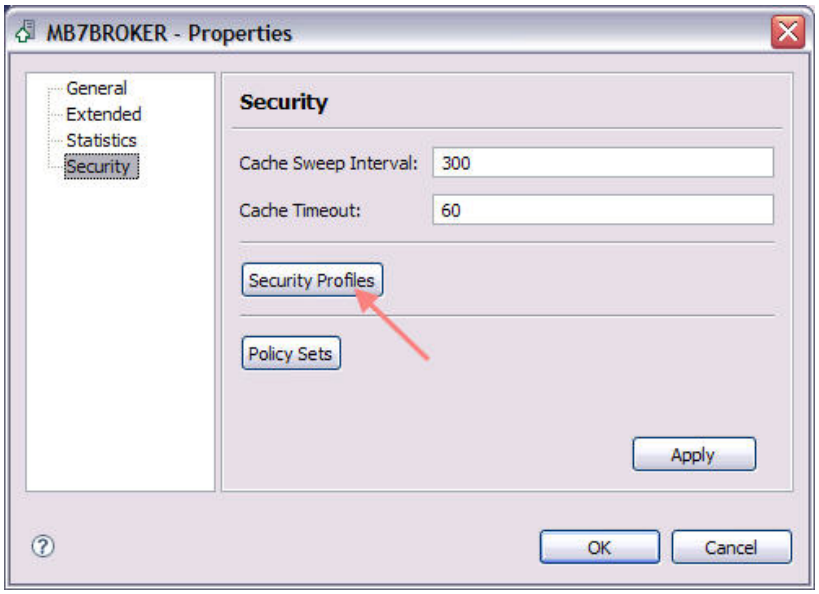
1. Open WebSphere Message Broker Explorer: right-click the **MB7BROKER** broker, and select **Properties**, as shown in Figure 4 below. Message Broker Explorer is an extension of WebSphere MQ Explorer with the add-on broker administration functions.

   Figure 4. Open broker properties on the WebSphere Message Broker Explorer



2. Select **Security** and then click **Security profiles**, as shown in Figure 5:

Figure 5. Broker security property



3. Click **Add** to add a new security profile, and rename it to LDAP_SP1. Fill in the fields with the values listed in Table 2 and Table 3, as shown in Figure 6 below. By using the editor in Message Broker Explorer, the value for Authentication Config is automatically generated from the combination of the LDAP host, LDAP baseDN, and LDAP uid attr. Similarly, the value for Authorization Config is automatically generated from the combination of LDAP host and LDAP group baseDN.

Figure 6. Add the security profile LDAP_SP1

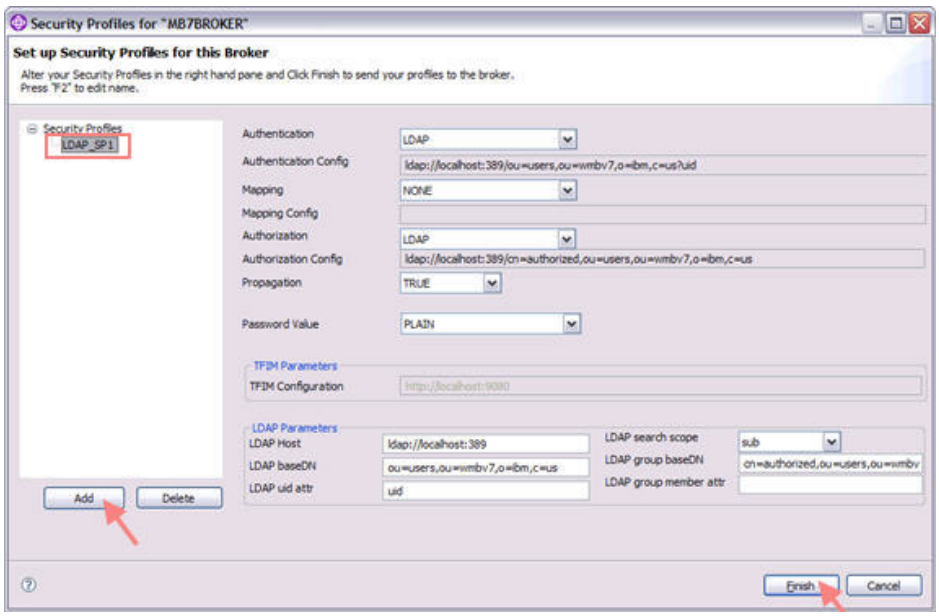


Table 3. Description of LDAP fields in the security profile

Field	Description	Value

LDAP host	define connection information to the LDAP server	ldap://localhost:389
LDAP baseDN	define base DN of all users to be authenticated	ou=users,ou=wmbv7,o=ibm,c=us
LDAP uid attr	define base DN's attribute to which the incoming user name maps	default value: uid
LDAP group baseDN	define base DN of the group in which users must be members to be authorized	cn=authorized,ou=users,ou=wmbv7,o=ibm,c=us
LDAP group member attr	define the attribute of the group used to filter the search	default value: member
LDAP search scope	define whether to perform a base or subtree search	default value: sub

Using the mqsicreateconfigurableservice command

You can also create a security profile using the mqsicreateconfigurableservice command on the Message Broker Command Console. Run the following command to create the security profile LDAP_SP2 based on the values listed in Table 2 above:

```
mqsicreateconfigurableservice MB7BROKER -c SecurityProfiles -o LDAP_SP2 -n authentication,
authenticationConfig,passwordValue,propagation,authorization,authorizationConfig -v "LDAP,
\"ldap://localhost:389/ou=users,ou=wmbv7,o=ibm,c=us?uid\",MASK,TRUE,LDAP,\"ldap://
localhost:389/cn=authorized,ou=users,ou=wmbv7,o=ibm,c=us\""
```

To list all security profiles on the MB7BROKER broker, use the following mqsireportproperties command:

```
mqsireportproperties MB7BROKER -c SecurityProfiles -o AllReportableEntityNames -r

SecurityProfiles
  Default_Propagation
    authentication='NONE'
    authenticationConfig=''
    authorization='NONE'
    authorizationConfig=''
    keyStore='Reserved for future use'
    mapping='NONE'
    mappingConfig=''
    passwordValue='PLAIN'
    propagation='TRUE'
    trustStore='Reserved for future use'
  LDAP_SP1
    authentication='LDAP'
    authenticationConfig='ldap://localhost:389/ou=users,ou=wmbv7,o=ibm,c=us?uid'
    authorization='LDAP'
    authorizationConfig='ldap://localhost:389/cn=authorized,ou=users,ou=wmbv7,o=ibm,c=us'
    keyStore='keystore.jks'
    mapping='NONE'
    mappingConfig=''
    passwordValue='PLAIN'
    propagation='TRUE'
    trustStore='Reserved for future use'
  LDAP_SP2
    authentication='LDAP'
    authenticationConfig='ldap://localhost:389/ou=users,ou=wmbv7,o=ibm,c=us?uid'
    authorization='LDAP'
    authorizationConfig='ldap://localhost:389/cn=authorized,ou=users,ou=wmbv7,o=ibm,c=us'
    keyStore='keystore.jks'
    mapping=''
```

```
    mappingConfig=''
    passwordValue='MASK'
    propagation='TRUE'
    trustStore='Reserved for future use'
```

As shown in the command output, in addition to LDAP_SP1 and LDAP_SP2, there is a security profile named Default_Propagation, which is a predefined profile that can be used for identity propagation by output/request nodes.

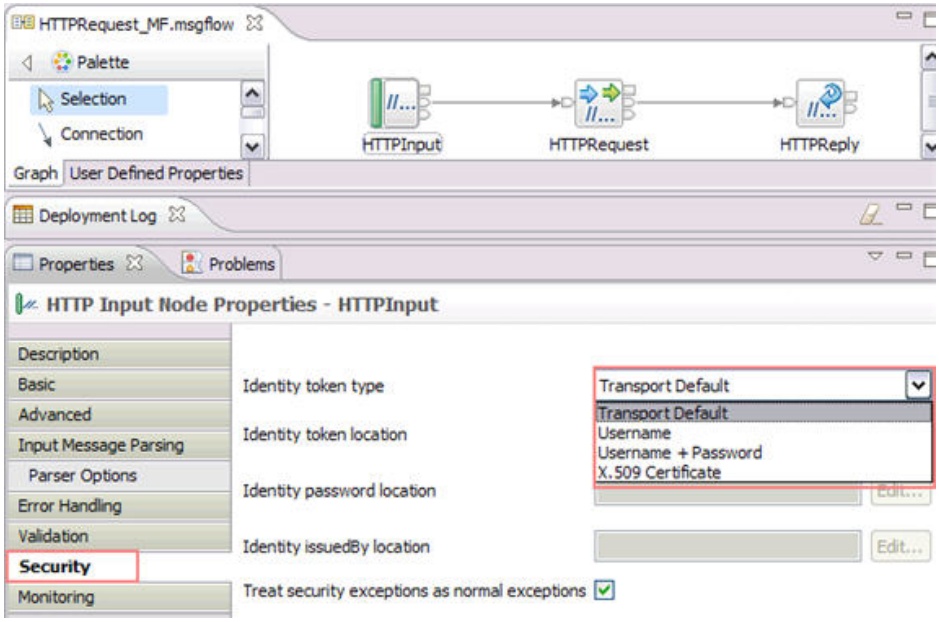
3. Configure identity extraction on message flows

Identity extraction can be configured on a message flow by selecting identity token type. The choice of the identity token type includes Transport Default, Username, Username + Password, and X.509 Certificate. If you select the Transport Default, user name and password are retrieved from transport level such as HTTP header. If the transport layer cannot provide the required identity credentials, such as the MQ MQMD header where only user name but not password can be contained, the identity information has to be taken from the incoming message body. If Username + Password is selected, you need to specify both the Identity token location and Identity password location. If X.509 Certificate is selected, the SSL is required and the user name is retrieved from the defined identity token location.

Obviously, the identity extraction can only apply to the input nodes. This article shows you how to configure both the HTTPInput node in the HTTPRequest_MF message flow, and the HTTPService node in the HTTPService_MF message flow for identity extraction:

1. Click the input node **HTTPInput** in the HTTPRequest_MF message flow. On the node's properties, select the **Security tab**.
2. Keep the default value Transport Default for Identity token type, as shown in Figure 7 below, so that the user name and password are extracted from the HTTP header.
3. Do the same for the node HTTPService in the HTTPService_MF message flow. The identity token type will be changed to Username + Password with the specified identity token location and identity password locations later during testing, as shown in Figure 16 below.

Figure 7. Select an identity token type



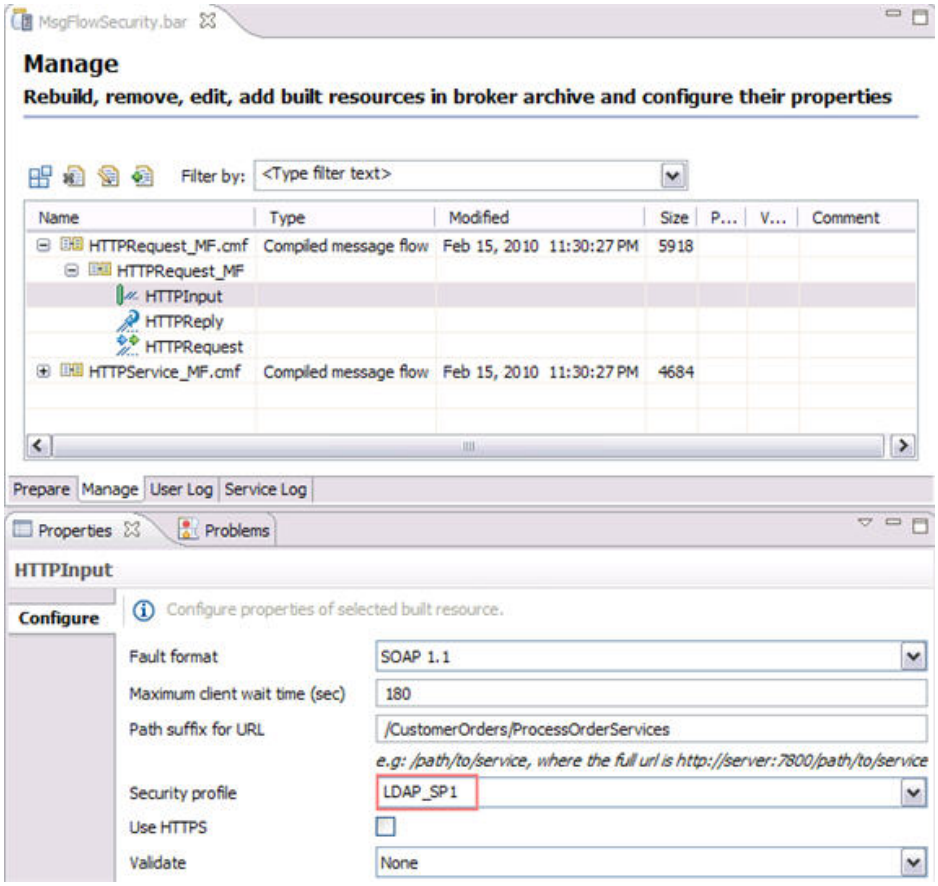
When a message flow is a Web service implemented using Message Broker SOAP nodes, the identity can be taken from the WS-Security tokens that are defined through appropriate policy sets and bindings.

4. Associate the security profiles with the message flows

To enable a message flow security, the input and output/request nodes need to be linked with a security profile that defines the security operations and the external security provider. Associate the security profiles with the input and request nodes in the message flows:

1. Create a BAR file named MsgFlowSecurity.bar and then add the message flows into it using the Toolkit.
2. On the **Manage** tab of the BAR file, expand the **HTTPRequest_MF** message flow and click the **HTTPInput** node. Type LDAP_SP1 in the Security profile field on the Properties of the message flow, as shown in Figure 8 below.
3. Follow the same steps to set the LDAP_SP2 security profile for the node HTTPRequest in the HTTPRequest_MF message flow and the node HTTPService in the HTTPService_MF message flow:

Figure 8. Set security profiles on the BAR file



Now each input and request node is associated with a security profile as listed in Table 4 below. You can set the security profile property at the message flow level, so that a node can inherit it if the node security profile property is left blank. Of course, a node security profile property can be set to No Security to explicitly turn off the security.

Table 4. Association of each input and request node with a security profile

Node	MessageFlow	SecurityProfile	Purpose
HTTPInput	HTTPRequest_MF	LDAP_SP1	authentication, authorization and password is in plain text
HTTPRequest	HTTPRequest_MF	LDAP_SP2	identity propagation and password is masked with ****
HTTPService	HTTPService_MF	LDAP_SP2	authentication, authorization and password is masked with ****

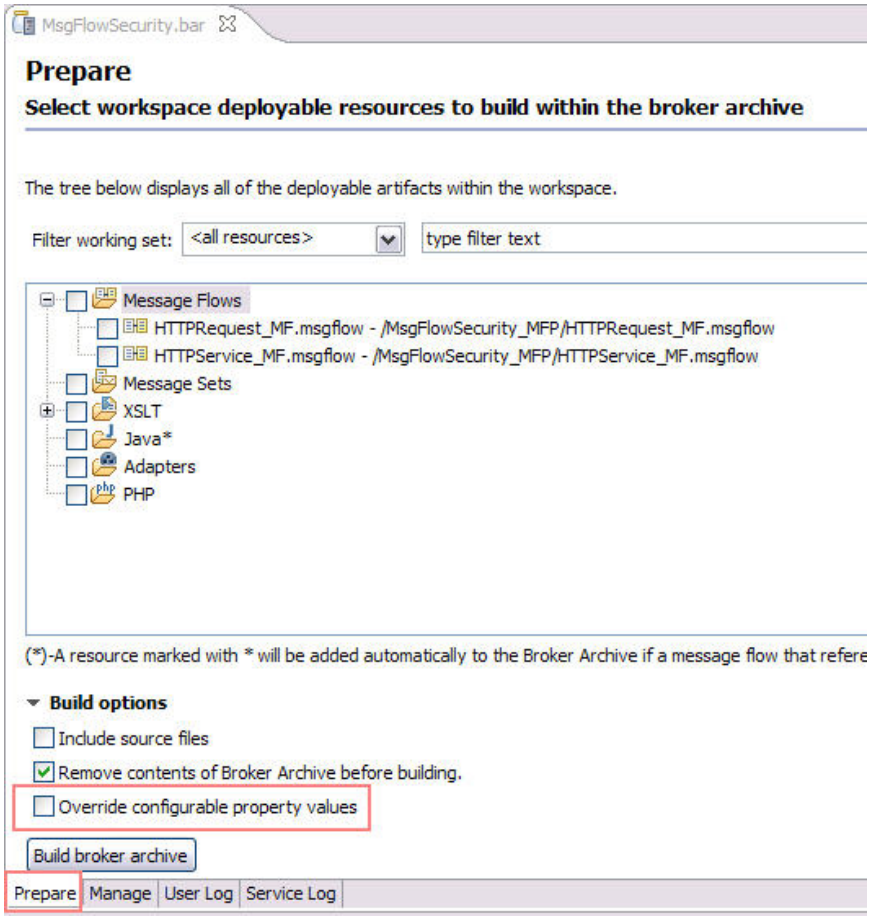
Message flow deployment

The Message Broker Toolkit is used to deploy the BAR file into the default execution group on the MB7BROKER broker (the default configuration of Message Broker V7). During the deployment process, the existence of the security profiles used in the message flows is checked. If the broker

cannot find any of the security profiles, the deployment process fails with an error indicating that some properties in the message flows are not recognized.

If you modify the message flows and rebuild the BAR file, you may see a warning message indicating that the configurable properties will be overridden, which means that the security profile property is about to be reset to the default value (blank). To change this behavior and ensure that no security profile name is lost, un-check Override configurable property values on the Prepare tab before building the BAR file:

Figure 9. Set build options on the BAR file



Message flow security testing

The flow debugger in the Message Broker Toolkit is used to track messages and catch any security exceptions. Breakpoints are set between nodes in the message flows for debugging. NetTool V4.7.2 is used as HTTP client to invoke the HTTPInput_MF message flow. NetTool is a free tool for debugging Web applications and Web services -- you can download it from SourceForge. You are going to test two scenarios: one for testing message flow security functions using the identity extracted from the HTTP header. and another extracted from the message body. Before conducting the tests, ensure that both the default execution group on the MB7BROKER broker and the LDAP server instance are running.

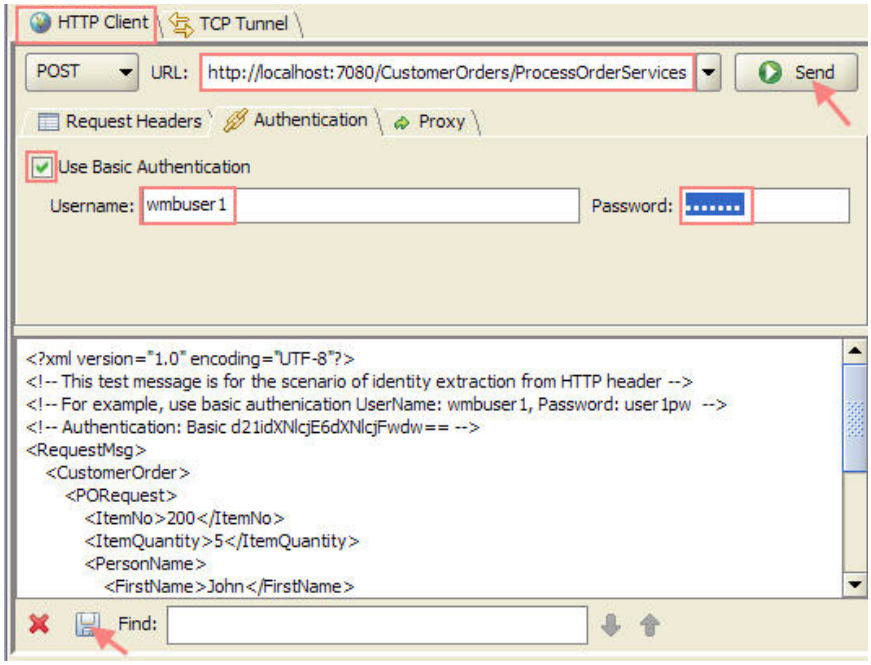
Examine security functions for identity extracted from the HTTP header

1. Open the Debug perspective on the Toolkit. Create a flow debugger instance, attach it to the default execution group, and then start the Flow Debugger.
2. Start NetTool and click the **HTTP client** tab. Type http://localhost:7080/CustomerOrders/ProcessOrderServices in the URL field and load the test file CustomerOrder1.xml, as shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This message is for test scenario of extracting identity from HTTP header -->
<!-- For example, use basic authentication UserName: wmbuser1, Password: user1pw  -->
<!-- Authentication: Basic d21idXNlcjE6dXNlcjFwdw== -->
<RequestMsg>
  <CustomerOrder>
    <PORequest>
      <ItemNo>200</ItemNo>
      <ItemQuantity>5</ItemQuantity>
      <PersonName>
        <FirstName>John</FirstName>
        <LastName>Doe</LastName>
      </PersonName>
      <Address>
        <Street>8501 IBM Dr</Street>
        <City>Charlotte</City>
        <ZipCode>28262</ZipCode>
      </Address>
    </PORequest>
  </CustomerOrder>
</RequestMsg>
```

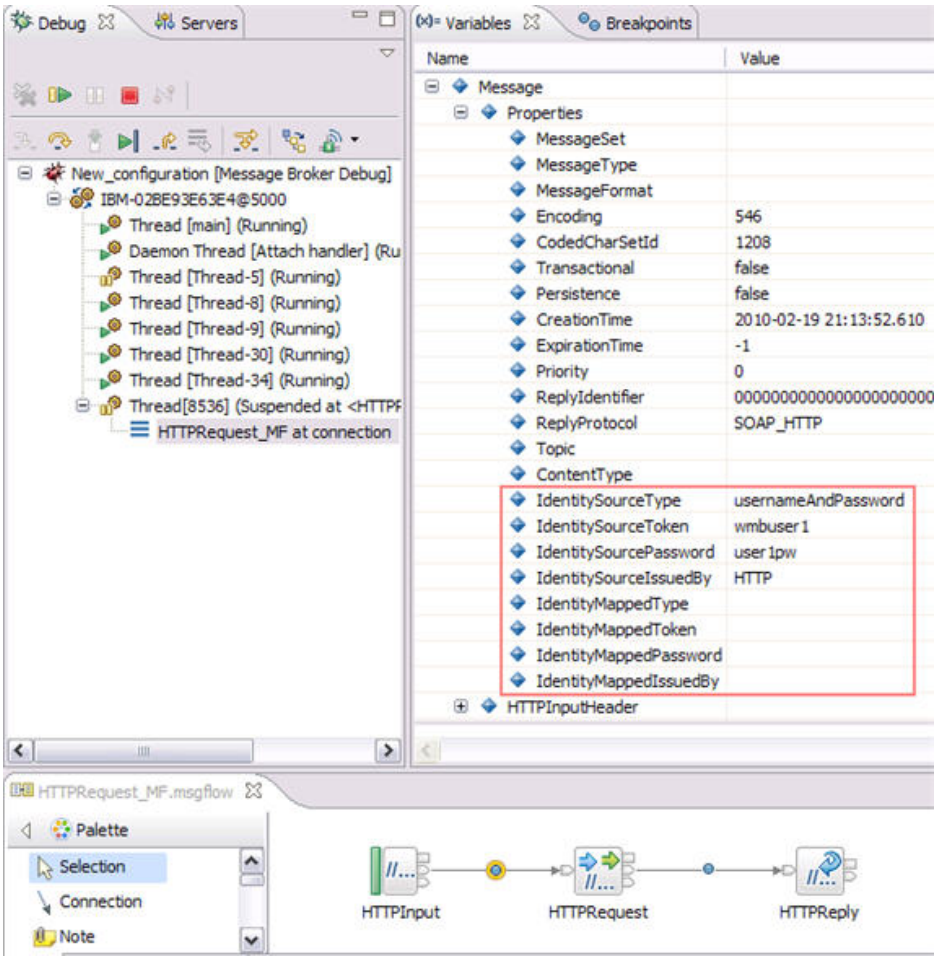
3. On the Authentication tab, check **Use Basic Authentication**, provide the username wmbuser1 and password user1pw, and click **Send** to invoke the HTTPRequest_MF message flow:

Figure 10. Set basic authentication on the NetTool HTTP client



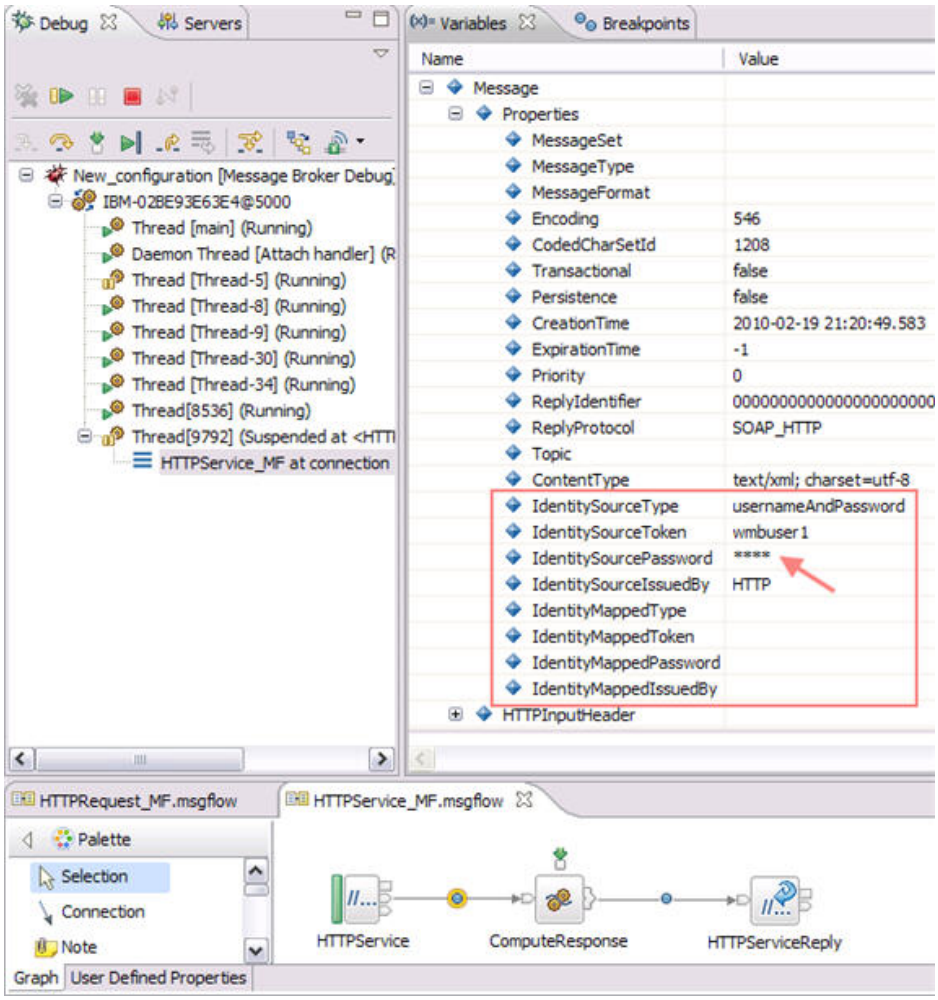
4. The pair of valid username and password from the HTTP header was successfully authenticated and authorized. As shown in the Flow Debugger, the identity information is extracted from the HTTP header into the Properties folder:

Figure 11. Extraction of identity information into the Properties folder



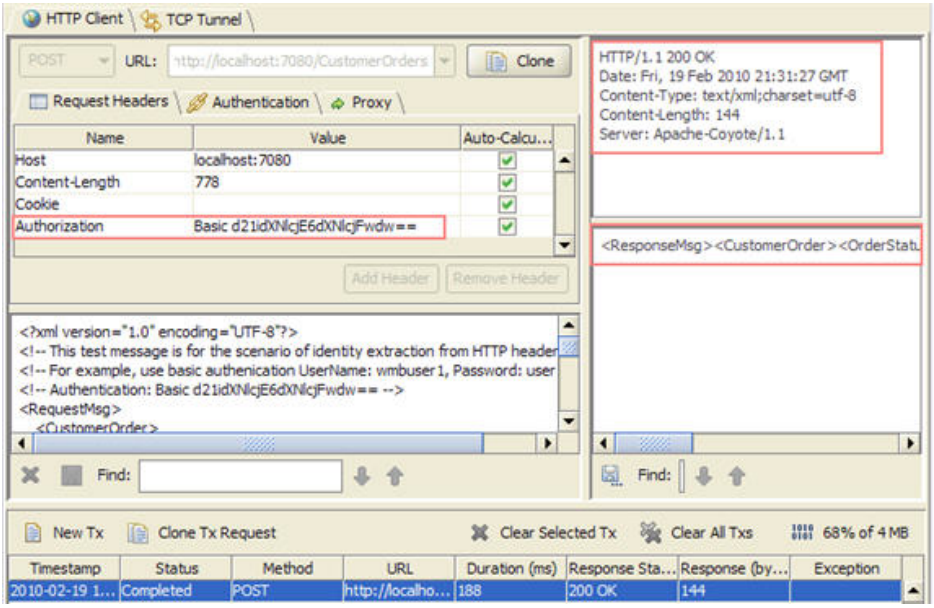
5. Proceed one step over on the Flow Debugger. The identity information is propagated in the HTTPRequest node and is passed to the HTTPService node in the HTTPService_MF message flow. As shown in Figure 12 below, the password is masked with ****, because the LDAP_SP2 security profile is used in the HTTPService node:

Figure 12. The masked identity password



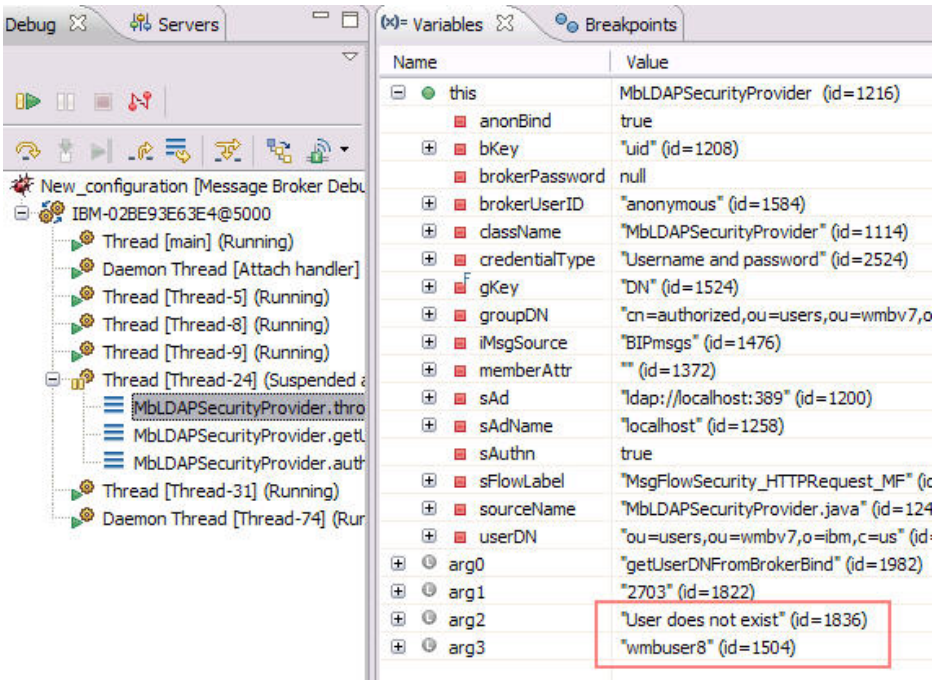
6. Complete the transaction. A success message is returned to NetTool:

Figure 13. Success message



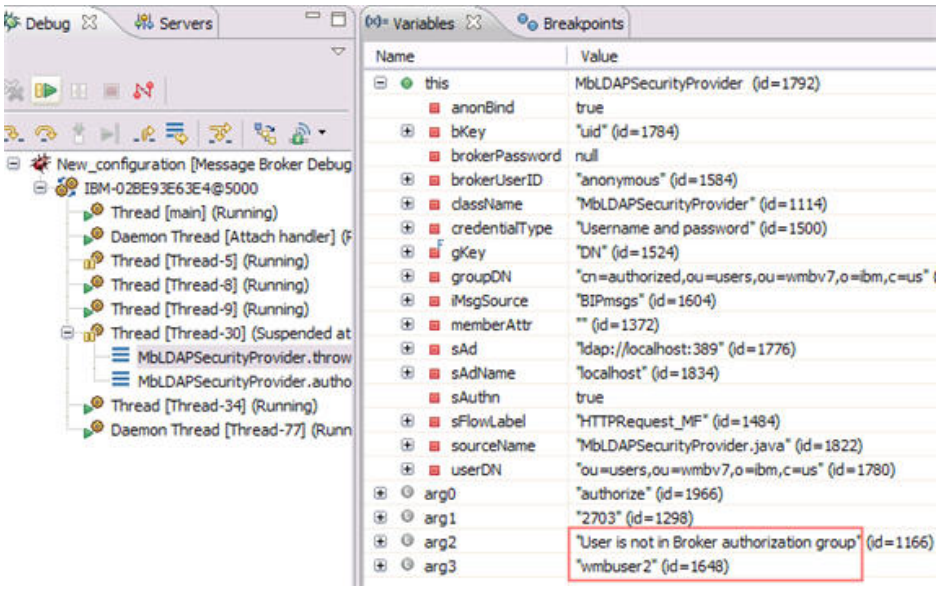
7. Start the test over from Step 3, and provide a non-existent username wmbuser8 and password to examine the authentication. Click **Send** to invoke the message flow.
8. On the Flow Debugger, an authentication exception is generated, indicating that the user wmbuser8 doesn't exist, as shown in Figure 14 below. Because of the authentication failure, access to the HTTPRequest_MF message flow is rejected:

Figure 14. User authentication error in the exception list



9. Start the test again over from Step 3 and provide a valid username wmbuser2 and password user2pw to examine the authorization. Click **Send** to invoke the message flow.
10. On the Flow Debugger, an authorization exception is generated, indicating that the user is not in the Broker authorization group, as shown in Figure 15 below, because wmbuser2 is not a member of the authorized group. Because of the authorization failure, access to the HTTPRequest_MF message flow is rejected:

Figure 15. An authorization error in the exception list

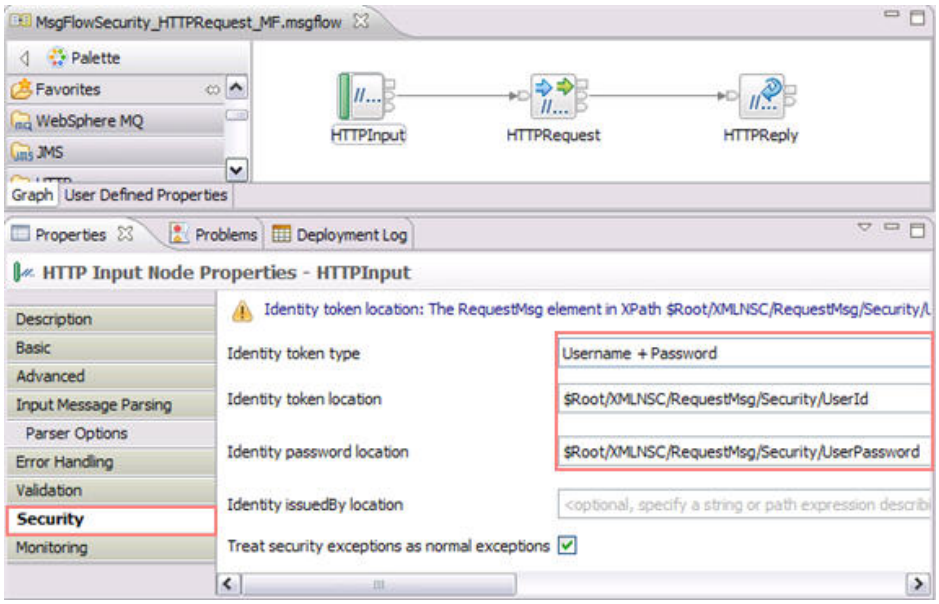


Examine security functions for identity extracted from the message body

To run this scenario, reconfigure the message flows for the identity token type change from Transport Default to Username + Password. That is, the identity is extracted from the message body instead of from the HTTP header.

1. In the Toolkit, open the HTTPRequest_MF message flow and click the **HTTPInput** node. On the Properties tab, click **Security**.
2. Select **Username + Password** from the drop-down list in the Identity token type field, and provide values in the Identity token location and Identity password location using XPath, as shown in Figure 16 below. Save the flow to complete the changes.
3. Repeat the previous steps to make the same changes on the **HTTPService** node in the HTTPService_MF message flow.
4. Rebuild and redeploy the BAR file with the message flows:

Figure 16. Configure identity extraction from the message body



5. Start the Flow Debugger if it is closed.
6. Start NetTool if it is closed. Load the test file CustomerOrder2.xml, as shown below. Click **Send** to invoke the message flow. The username wmbuser1 and its password user1pw are provided in the message body:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This message is for test scenario of extracting identity from the message body -->
<RequestMsg>
  <!-- The UserId and UserPassword need to be changed for various tests-->
  <Security>
    <UserId>wmbuser1</UserId>
    <UserPassword>user1pw</UserPassword>
  </Security>
  <CustomerOrder>
    <PORequest>
      <ItemNo>200</ItemNo>
      <ItemQuantity>5</ItemQuantity>
      <PersonName>
        <FirstName>John</FirstName>
        <LastName>Doe</LastName>
      </PersonName>
      <Address>
        <Street>8501 IBM Dr</Street>
        <City>Charlotte</City>
        <ZipCode>28262</ZipCode>
      </Address>
    </PORequest>
  </CustomerOrder>
</RequestMsg>
```

7. From the Flow Debugger, the identity information is displayed on the Properties folder, which is the same as shown in Figure 11 above, but the identity information is extracted from the message body.
8. As expected, a successful response is returned to NetTool, since the correct username and password were provided.

9. You can modify the test message CustomerOrder2.xml with invalid values for UserId and UserPassword, and then invoke the message flow through NetTool following the same steps to examine the authentication and authorization security functions.
10. In addition, you can use the cURL command tool to invoke the HTTP calls. You can download cURL V7.19.7 for free from Softpedia. Run the following command with the username wmbuser3 and password user3pw in the test message CustomerOrder2.xml. A successful response message is returned, because the correct credentials were provided within the message body. The wrong username wmbuser8 and password user8pw provided in the command parameter are ignored by the security manager, since the identity token type is Username + Password:

```
curl --basic --data-binary @CustomerOrder2.xml -u wmbuser8:user8pw
http://localhost:7080/CustomerOrders/ProcessOrderServices

<ResponseMsg>
   <CustomerOrder>
      <OrderStatus>Available</OrderStatus>
      <ItemNo>200</ItemNo>
      <ItemQuantity>5</ItemQuantity>
   </CustomerOrder>
</ResponseMsg>
```

11. Run the following command with a non-existent username wmbuser8 and password user8pw in the test message CustomerOrder2.xml. The error `This server could not verify that you are authorized to access the document requested" is returned`:

```
curl --basic --data-binary @CustomerOrder2.xml
http://localhost:7080/CustomerOrders/ProcessOrderServices

<html>
  <head>
    <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <titleg>401 Authorization Required</title>
  </head>
  <body>
    <h1>401 Authorization Required</h1>
    This server could not verify that you are authorized to access the document requested.
  </body>
</html>
```

Conclusion

WebSphere Message Broker V6.1 or later provides a security manager for implementation of message flow security, so that the end-to-end processing of a message through a message flow is secured based on an identity carried in that message instance. To configure the message flow security, an external security provider is required, such as Tivoli Directory Server, to be used as the security enforcement. A security profile (or multiple security profiles for different security requirements) needs to be created to specify what security functions need to be performed (such as authentication, authorization, and identity propagation), as well as which external security provider is used. The message flows also need to be configured on how the identity can be extracted from input nodes, and which security profile needs to be associated with input and output/request nodes. The example in this article has shown you how to implement the security configuration.

Downloadable resources

Description	Name	Size
WebSphere Message Broker Project Interchange file	MsgFlowSecurity_PI.zip	10 KB

Related topics

- Security identity propagation sample
  This sample from the WebSphere Message Broker V7 information center shows you how to extract security credentials from messages received at the MQInput and HTTPInput nodes, how to manipulate these security credentials using ESQL, and how to propagate the identity to the MQOutput and HTTPRequest nodes.
- WebSphere Message Broker product page
  Product descriptions, product news, training information, support information, and more.
- WebSphere Message Broker V7 information center
  A single Web portal to all WebSphere Message Broker V7 documentation, with conceptual, task, and reference information on installing, configuring, and using your WebSphere Message Broker environment.
- What's new in WebSphere Message Broker V7
  WebSphere Message Broker V7 provides universal connectivity with its ability to route and transform messages from anywhere to anywhere. Through its simple programming model and a powerful operational management interface, it makes complex application integration solutions much easier to develop, deploy, and maintain. This article describes the major enhancements in V7.
- Download free trial version of WebSphere Message Broker V7
  WebSphere Message Broker V7 is an ESB built for universal connectivity and transformation in heterogeneous IT environments. It distributes information and data generated by business events in real time to people, applications, and devices throughout your extended enterprise and beyond.
- WebSphere Message Broker documentation library
  WebSphere Message Broker specifications and manuals.
- Redbook: Patterns: SOA design using WebSphere Message Broker and WebSphere ESB
  Patterns for e-business are a group of proven, reusable assets that you can use to more quickly develop and deploy e-business applications. This Redbook shows you how to use WebSphere Message Broker with WebSphere ESB to implement an ESB within an SOA. Includes a scenario to demonstrate design, development, and deployment.
- developerWorks on Twitter
  Check out recent Twitter messages and URLs.