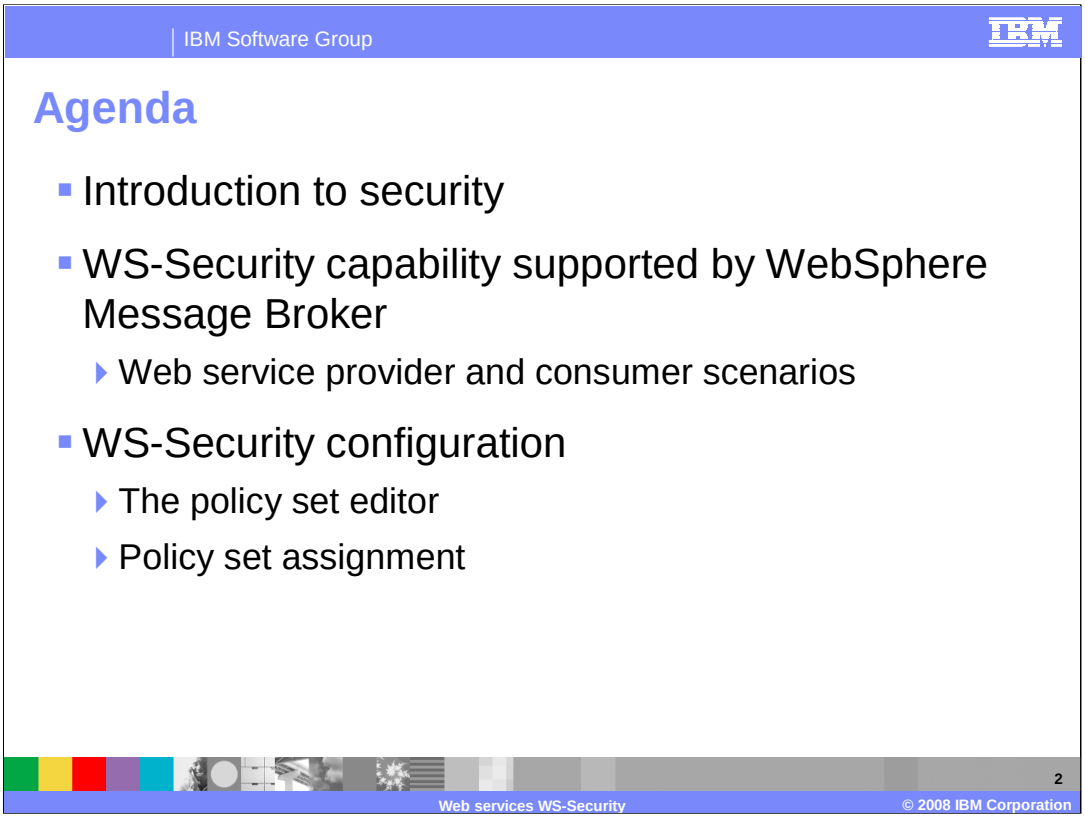
This presentation will discusses the Web services security function in WebSphere Message Broker Version 6.1.

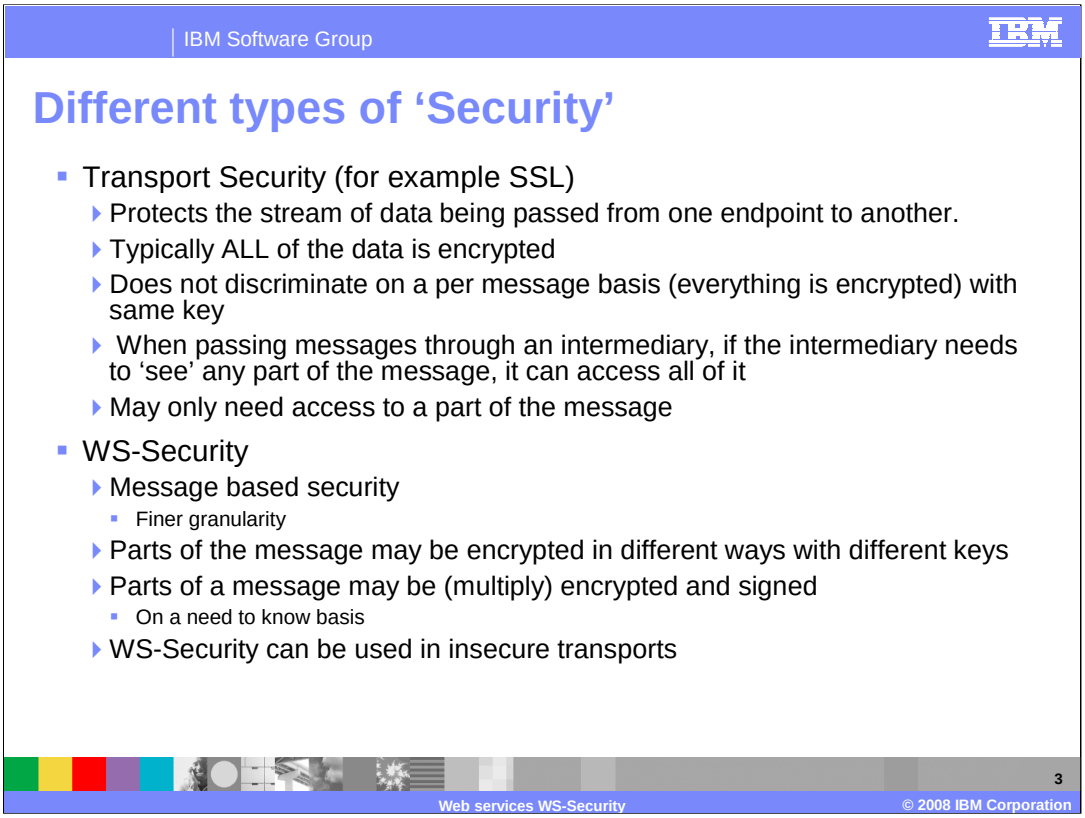


This presentation will discuss three key areas.

First, the presentation will provide a recap of general security requirements, and will relate this to a Web services environment.

It will then discuss the WS-Security provided in Message Broker version 6.1, and will show how this is used with the new SOAP Nodes. It will then apply this to two scenarios, the provider scenario and the Consumer, or Client scenario.

Finally, it will discuss how to configure the WS-Security settings, and show how the Policy Set Editor is used for this function.



There are many ways of configuring and using security. For example transport security protects all of the data transmitted over a particular channel. All of the data being transmitted over that channel is treated in the same way, and individual messages are not treated differently. If encryption is used, then this applies to the whole message.

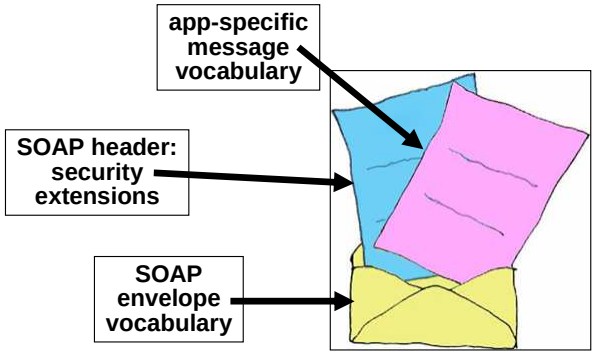
If an intermediary is used to process the data, then the entire message must be decrypted, so that the intermediary can access the message. It is not possible to decrypt only a part of the message.

Conversely, WS-Security is designed to protect data at the individual message level. This provides a much finer granularity of security than transport security.

WS-Security allows you to encrypt only a part of the message, or to encrypt parts of the message with different keys. It is possible to use WS-Security over a transport that is itself insecure, for example if SSL is not used at the transport level.

Review: SOAP message structure

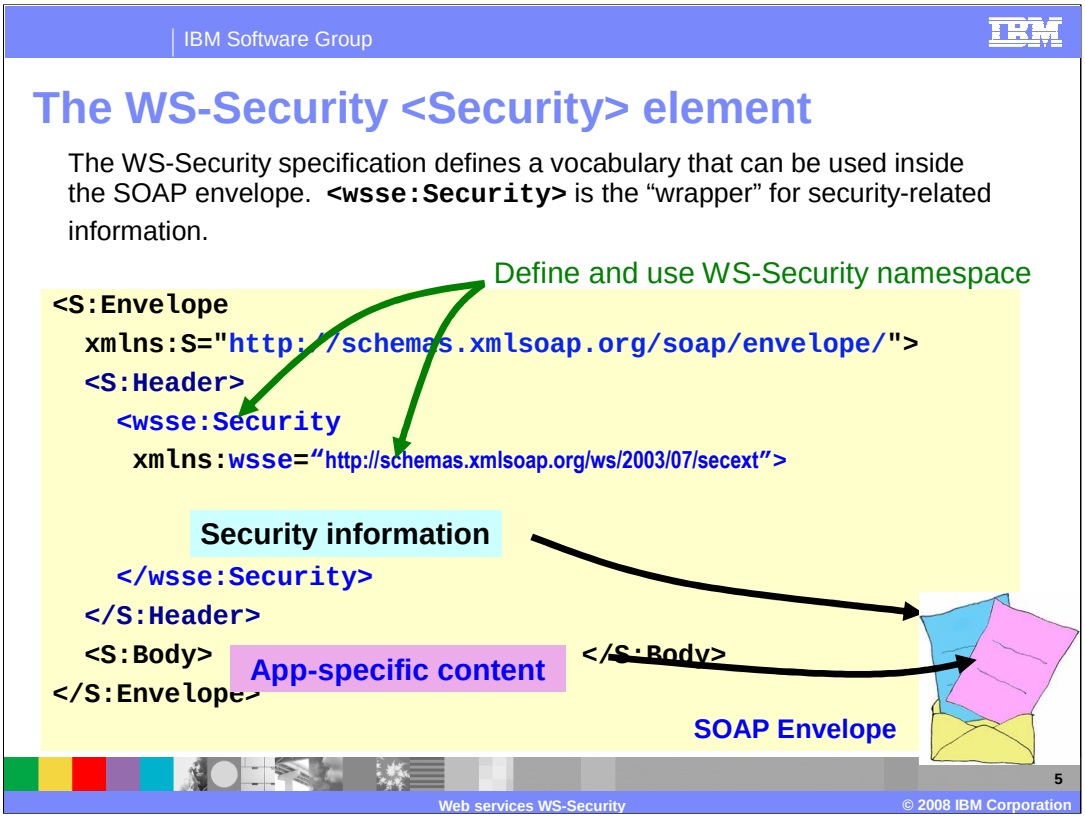
- The SOAP specification defines the “envelope” vocabulary
  - The "envelope" wraps the message itself
- WS-Security defines the <Security> element, which allows security extensions to be placed in <soapenv:header>
  - Username/password
  - X.509 certificate
  - Encryption details
  - XML Signature



A SOAP message consists of an envelope, which contains the whole message. Within the envelope is the actual message, which consists of header information and a SOAP body. The body is the real payload, the message which an application program will typically need to access.

WS-Security defines a new Security element which may be placed in the header section of a SOAP message.

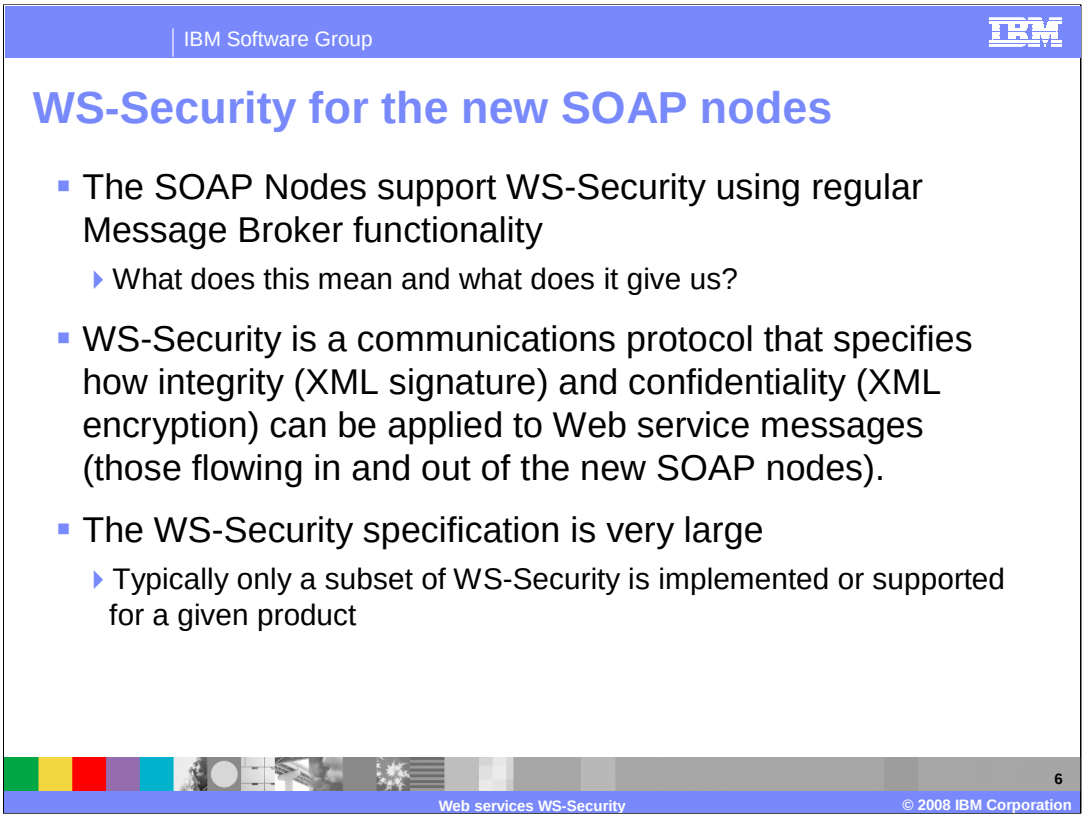
There are many considerations concerning security with SOAP messages. The task is to keep the message secure through intermediate systems until it reaches the ultimate recipient.



The example shows a SOAP message with a skeleton showing where WS-Security should be placed.

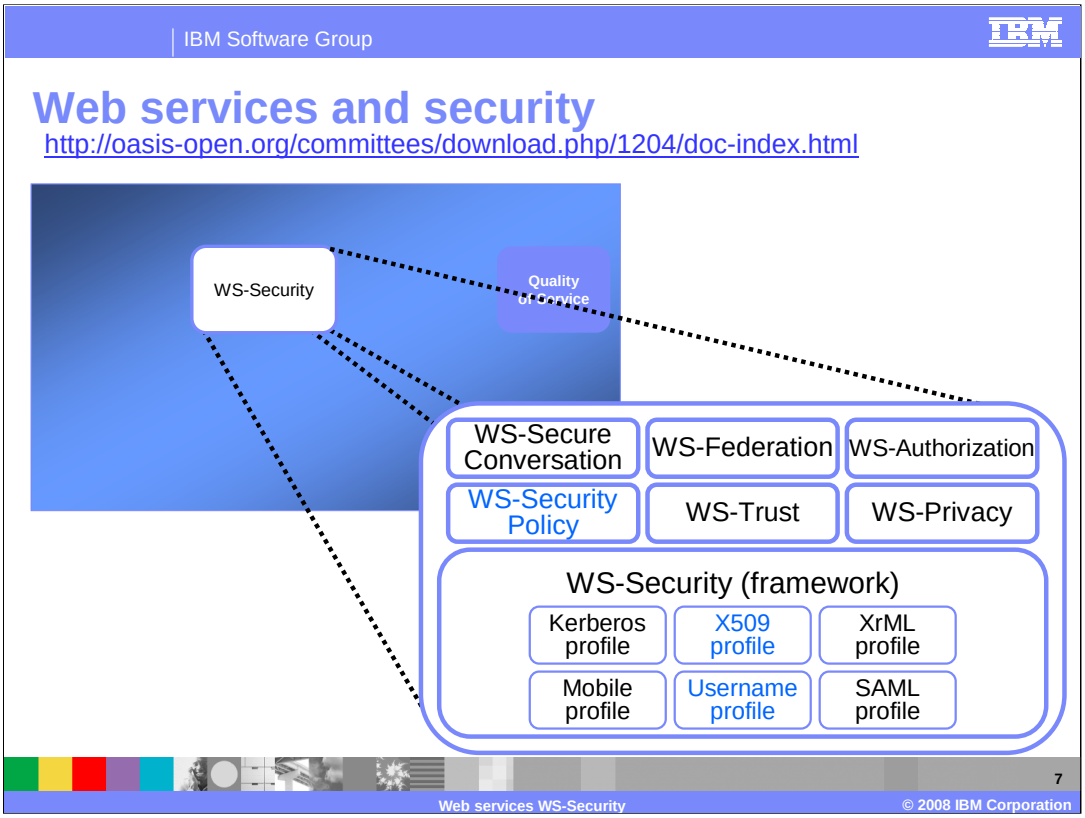
At the top level is SOAP Envelope, then inside it is the SOAP Header and the Body.

The WS-Security specification defines a set of elements which may be used inside a SOAP envelope. The element "wsse:Security" is used to contain the security information.



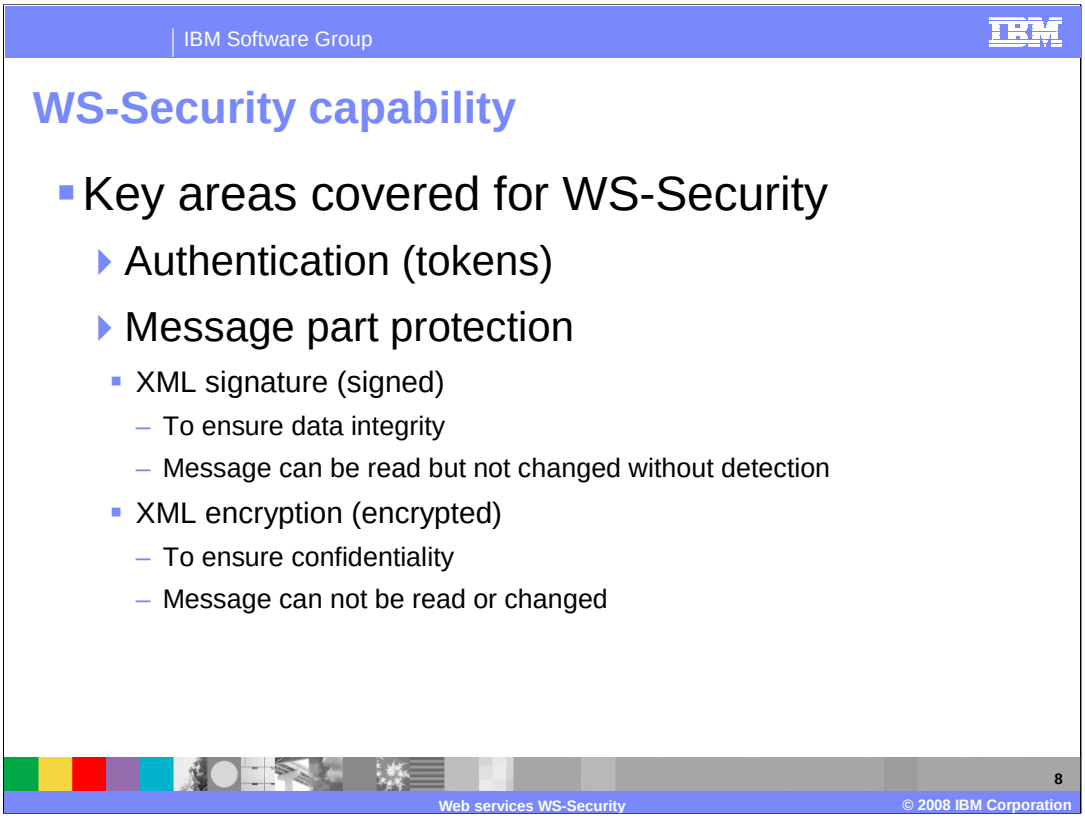
Message Broker version 6.1 supports WS-Security using the SOAP nodes. This is implemented by using the properties of the SOAP Input or SOAP Request Nodes.

It is important to note that the WS-Security specification is very large, and many software vendors do not implement the whole of the specification. A software component will normally implement the security features that are appropriate for the type of security usage that will be required.



What is often thought of as a single specification, WS-Security, is actually a large set of interrelated specifications. The slide shows the components which constitute WS-Security.

Message Broker version 6.1 focuses on the part of the WS-Security specification known as WS-Security Policy.

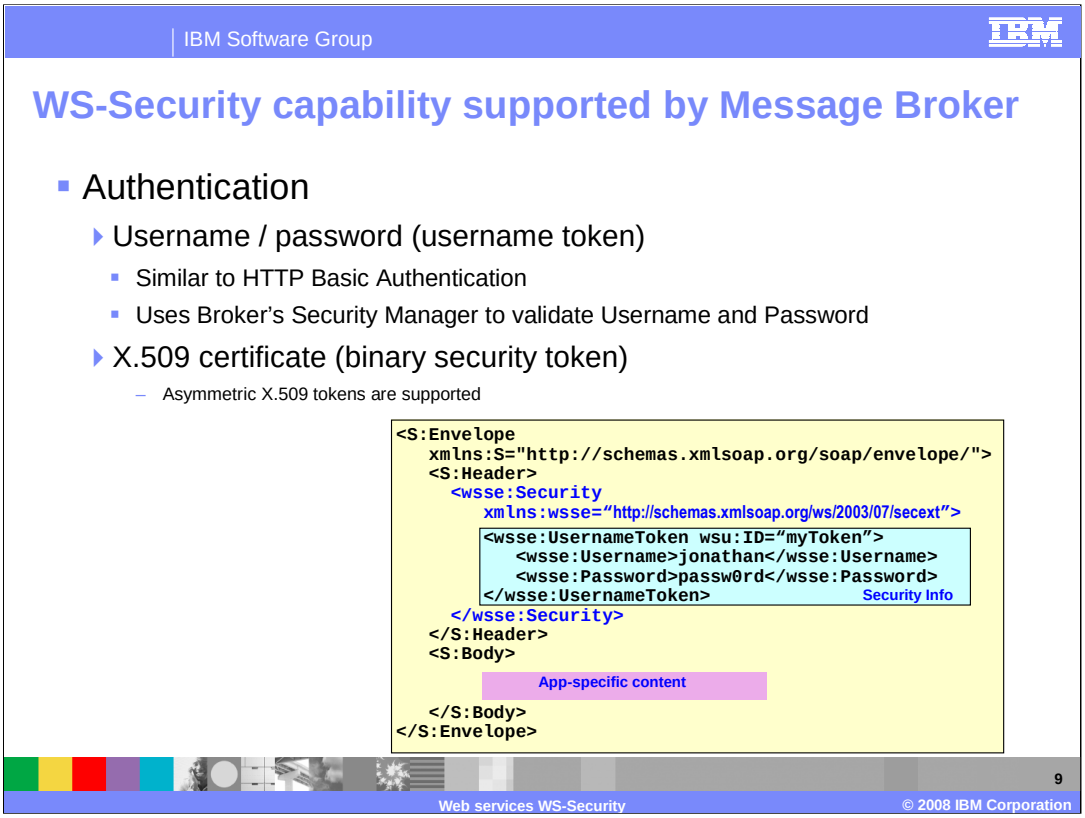


The key components of the WS-Security implementation within Message Broker version 6.1 are authentication, and message protection.

Authentication of the request will be achieved using a certificate or token.

Message protection can be achieved using either a digital signature, or using encryption. If a message part is signed, this means that the part can be read, but not changed. This ensures data integrity of the message. If the message is changed by an intermediary, then the final recipient of the message can detect this change, and take appropriate action. An example of this usage would be an order for goods place over the internet. This data may be visible to anyone, but it's important that the data is not changed.

If the message is encrypted, then the message can not be read or changed. An example of this usage would be to protect the credit card details of the person who placed the order for goods.



This slide discusses the Authentication capability provided in Message Broker for WS-Security.

The types of security tokens that are defined by Web services security are User-name token and Binary security token .

A user-name token consists of a user-name and, optionally, password information. You can include a user-name token directly as an element in the Security header within the SOAP message, rather than the HTTP header. Binary tokens, such as X.509 certificates, **Kerberos** tickets, Lightweight Third Party Authentication tokens, or other non-XML formats, require a special encoding for inclusion. The Web services security specification describes how to encode binary security tokens such as X.509 certificates and Kerberos tickets, and it also describes how to include opaque encrypted keys. The specification also includes extensibility mechanisms that you can use to further describe the characteristics of the credentials that are included with a message. The User-name Token is used to provide a user-name within the SOAP message. It is used as part of the basic authentication process.

The Broker Security Manager must be enabled, and a security profile specified. If not all user-names and passwords will be accepted.

Message Broker supports binary tokens for authentication. The tokens that are supported are asymmetric X/509 tokens.

Symmetric X.509 tokens are not supported.

The example shows a "User-Name Token" being specified. Note that the user-name and password are specified in clear text. This may be encrypted using message part encryption. This is described later.

WS-Security capability supported by Message Broker

- Message part protection (whole body)
- Whole body can be:
  - Encrypted (XML encryption)
  - Signed (XML signature)
  - Can be signed and encrypted if required
    - Using different certificates

```
<S:Envelope>
  <S:Header>
    <wsse:Security S:mustUnderstand="1"
       xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/07/secext">
      <wsse:BinarySecurityToken EncodingType="wsse:Base64Binary">
         MIIDQTCC4ZzO7tIgerPlaid1q ... [truncated]
      </wsse:BinarySecurityToken>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      ....signature data....
      </ds:Signature>
    </wsse:Security>
 </S:Header>

 <S:Body>
    <m:OrderAircraft quantity="1" type="777" config="Atlantic"
       xmlns:m="http://www.boeing.com/AircraftOrderSubmission"/>
 </S:Body>
<S:Envelope>
```

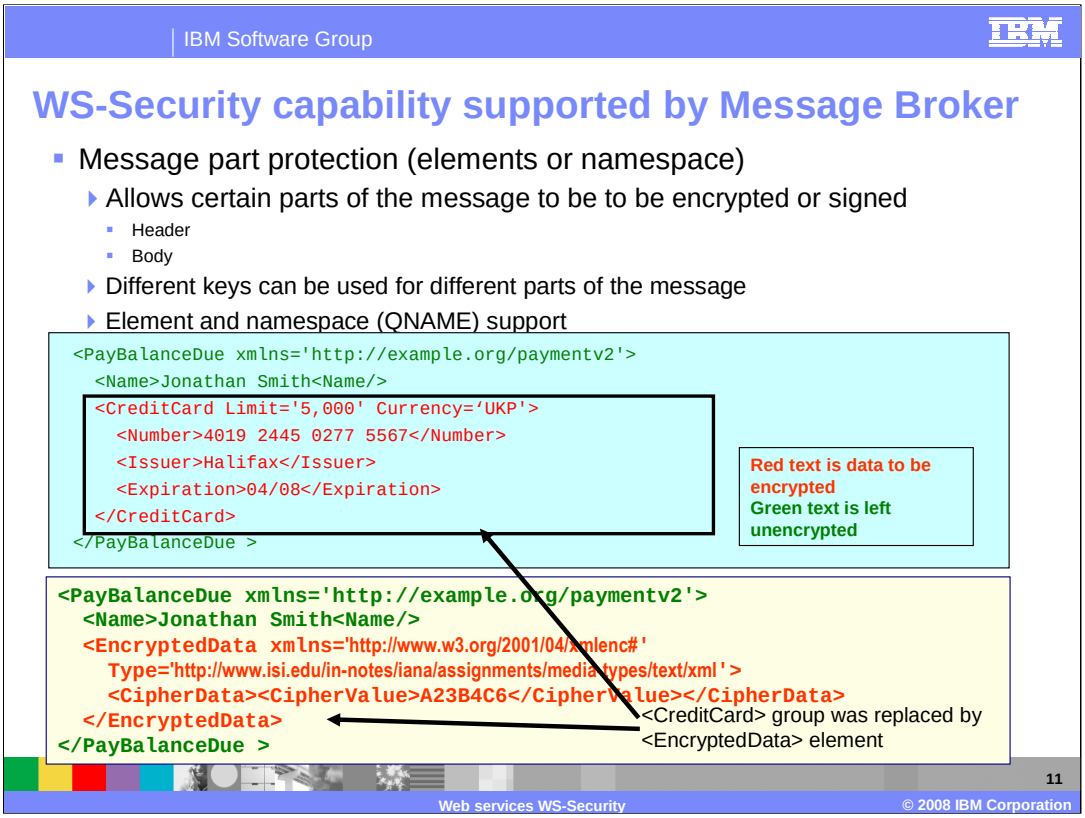

Message Broker also supports message protection. This can either be the whole message, or part of the message.

The message can be encrypted, or digitally signed, or both.

These different security functions can be used using different security certificates, and can be used separately from each other.

The example shows the payload being digitally signed. However, it is not encrypted, and is therefore still readable by intermediaries.

Using this technique, you can allow intermediaries to have access to certain parts of the message, but inhibit access to other parts of the message.



As discussed earlier, it is possible to protect parts of the SOAP messages, rather than the entire message. This is done by specifying elements and namespaces within the header or body of the SOAP message that you want to sign or encrypt. Namespaces are specified using the QNAME function.

Because message part protection can specify the header, it can be used in conjunction with User-name tokens to protect the user-name and password.

This slide shows an example of how this might be used. The example shows a credit card transaction, where the credit card details, shown in red, have been encrypted. The section at the bottom of the slide, shown in yellow, contains the message that is sent over the transmission channel. The sensitive credit card data is shown as encrypted.

Note that the encrypted information this example has been truncated.

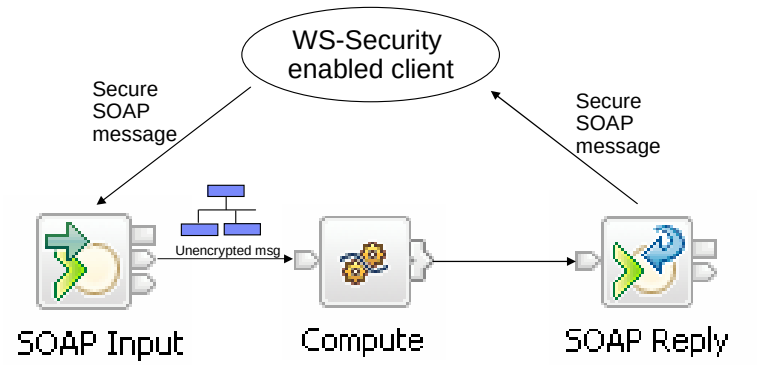
Section

Web service provider and consumer scenarios

The next three slides show the three primary scenarios for Web Services within Message Broker. These are the Web Service provider scenario, and the synchronous and asynchronous Web Service client scenarios.

Web service provider scenario

- The WS-Security function is handled by the SOAP nodes



- SOAP input node authenticates, decrypts and verifies the signature of the message and message parts
- Message passes unencrypted through the message flow
- SOAP reply node encrypts and signs the appropriate parts of message

This slide shows the scenario where the message flow represents the Web service. A Web service client invokes the message flow by sending it a SOAP message. In this case, the SOAP message is secured using WS-Security.

The role of the SOAP Input node is to verify all signed parts and decrypt the message as defined within the configuration. It will do this using the security configuration that it has been given.

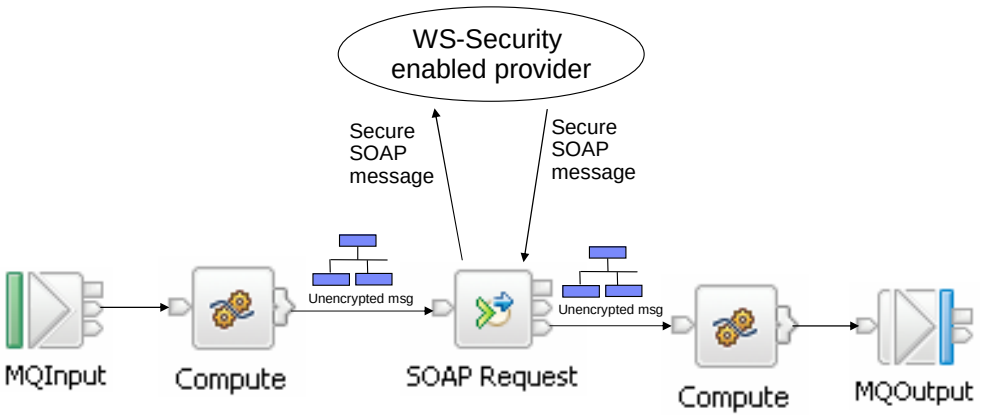
The role of the SOAP Reply node is to apply any required WS-Security function.

If the WS-Security configuration is applied to the whole message, then this is decrypted for use by the message flow.

If the WS-Security configuration defines parts of the message, then the SOAP input node decrypts these parts for use by the message flow. Any parts of the message that Message Broker is not allowed to see, since it doesn't contain the correct certificates, remain encrypted.

When the message reaches the SOAP Reply node, the message must be signed and encrypted, as defined by the WS-Security configuration for the message flow.

Web service client scenario



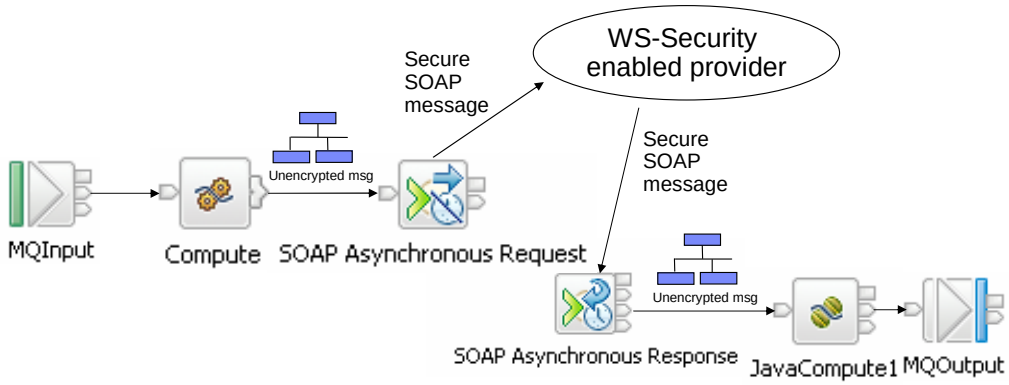
- SOAP request node encrypts and signs the appropriate parts of message
- Message is sent to the WS-Security enabled provider
- SOAPrequest node decrypts and verifies the signature of the message and message parts

In the Web service client scenario, the SOAP Request node is responsible for creating the outbound SOAP message with the appropriate security credentials.

The message is unprotected as it is given to the SOAP Request node. The SOAP request node encrypts the appropriate parts of the message, and adds any necessary digital signatures. It then sends the secure message to the Web Service provider, which will be configures to receive the secure message.

When this is returned, the SOAP Request node receives the secure response, and performs the same function as the SOAP Input node, before passing the message to the rest of the message flow.

Web service client scenario - Asynchronous



- SOAP AsyncRequest node encrypts and signs the appropriate parts of message
- Message is sent to the WS-Security enabled provider
- SOAP AsyncResponse node decrypts and verifies the signature of the message and message parts

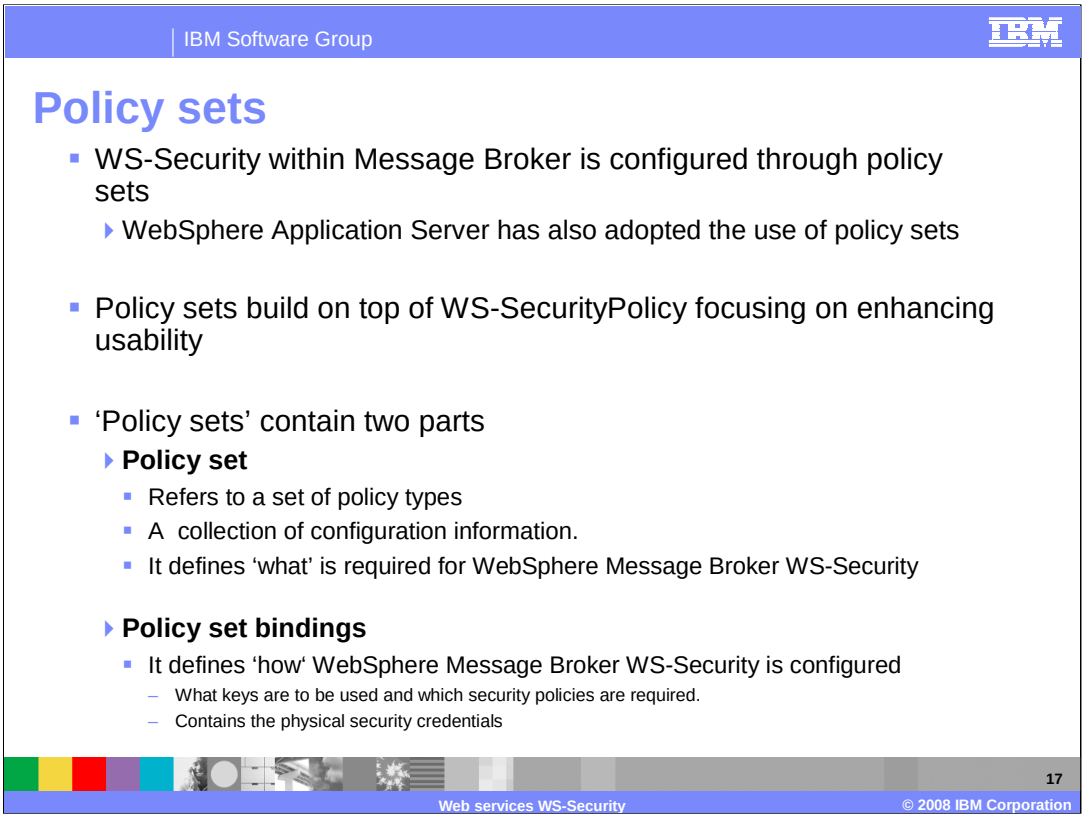
The third scenario is the asynchronous case, which operates in a similar way to the synchronous SOAP Request node.

The SOAP Asynchronous Response node again operates in the same way as the SIOAP Input node, and processes the returned WS-Security configuration.

Section

WS-Security configuration

The second part of this presentation discusses how to configure WS-Security within the Message Broker environment.



Configuration of WS-Security is done using Policy Sets.

Policy Sets builds on WS-Security-Policy, and adds several usability components to the base function.

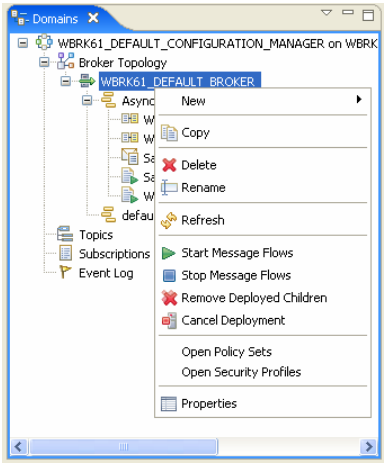
To ensure a common security implementation across the WebSphere set of products, both WebSphere Application Server and WebSphere Message Broker use Policy Sets to perform this function.

A Policy Set refers to a set of policy types. Each Policy Set contains two parts. The Policy Set itself refers to a set of policy types. Message Broker supports only the WS-Security policy type. The policy set defines what is needed for WS-Security; it defines the different types of authentication, integrity and confidentiality that are needed. And it defines which parts of the message need to be signed or encrypted.

The policy set bindings define how the security policies are to be used and applied to the defined message. It contains the physical security credentials, for example which keys are going to be used, which certificates are going to be used, where are these certificates stored.

Launching the policy set editor

- The existing broker administration view has a new option when right-clicking on a broker to “open policy sets”
- Selecting “Open Policy Sets” will open the policy set editor



The policy sets and policy set bindings are defined at the broker level.

The policy set editor is launched by right clicking on the broker and selecting ‘Open Policy Sets’.

Policy set editor

- A default policy set and policy set binding are always present
  - WSS10Default
  - This can not be edited or deleted
  - Configured for user name authentication token
- Policy sets and policy set bindings are defined at the broker level

Opening the Policy Set Editor will show a window similar to the one shown on this slide.

A Default Policy Set and binding are created called “WSS10 Default”. This contains default settings, for example to encrypt the whole message.

This Policy Set and Policy Set Binding name is also used within WebSphere Application Server and contains the same default settings. Whenever you create a broker, this default policy set is created as part of that process.

Although you can use the default policy Set and Policy Set Binding, it is not editable. If you need to change the settings provided with the default Policy Set, then you need to create your own policy set. This can be created from scratch, or you can copy and past the default policy set into your own policy set. You can also copy and paste your own policies. Changes can then be made to the new policy set.

Creating and deleting policy sets and bindings

- Once created, policy sets and policy set bindings can be configured and modified
- A policy set binding must refer to a previously created policy set
  - Different policy set bindings can refer to a single policy set
- You can not delete a policy set if one or more policy set bindings refer to it
- The policy set editor allows the administrator to add, delete, import and export a policy set and policy set binding
- A ‘deploy’ is not required
  - Clicking on Finish updates the broker with the new configuration
  - Does not follow the deployment mechanism

The policy set binding must refer to a policy set that has been previously created. This can be done by creating new policy sets and bindings, or you can associate a new policy set binding with an existing policy set. You can associate multiple bindings with the same policy set. You cannot delete a policy set until all associated bindings have been deleted.

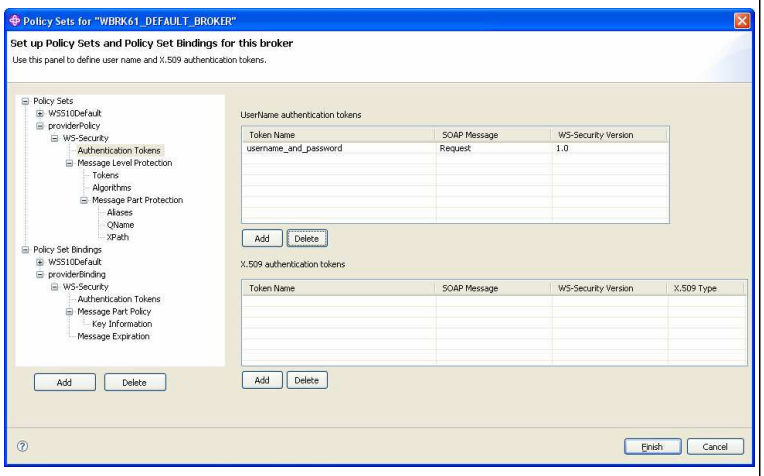
The policy set editor does not deploy Policy Set and Binding information. When you click Finish in the policy set editor, the new policy is sent directly to the broker using the Configuration Manager.

Using the policy set editor, you can edit, save, import and export a policy set.

This can be used export a Policy Set for use on another broker instance, perhaps located on a separate system. This policy can be imported into another system using the command line, which provides an alternative to the policy set editor.

Policy set configuration overview

- Includes the capability to configure
  - Authentication tokens
    - Username and password
    - x.509 certificates
    - *Identity*
  - Message part protection
    - XML signature and XML encryption
      - *Integrity and confidentiality*
    - Can assign different keys to different parts of the message
      - Can encrypt multiple times
- Defines security requirements for inbound **and** outbound message



When defining the policy, it is important to remember the inbound and outbound messages may differ. The Policy Set defines which parts of the message need to be encrypted and signed for both parts. These are referred to as the 'Request' and the 'Response'. These could use different tokens and different certificates.

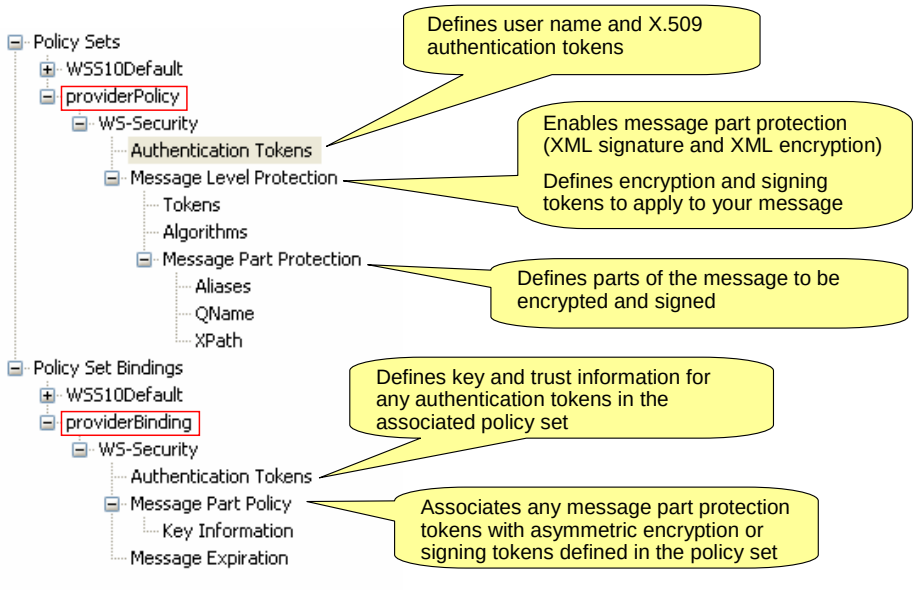
If you want to define Message Level Protection, then you can specify the Signature, and the type of encryption. If you specify just signature, then the data can be seen intact, but not changed. If you specify Encryption, then the data cannot be read.

For Message Part Protection, you can again specify Signature and Encryption, in the same way as in Message Level Protection. In this case, this can be applied to individual parts of the message. Certain parts may be signed, whereas other parts may be encrypted.

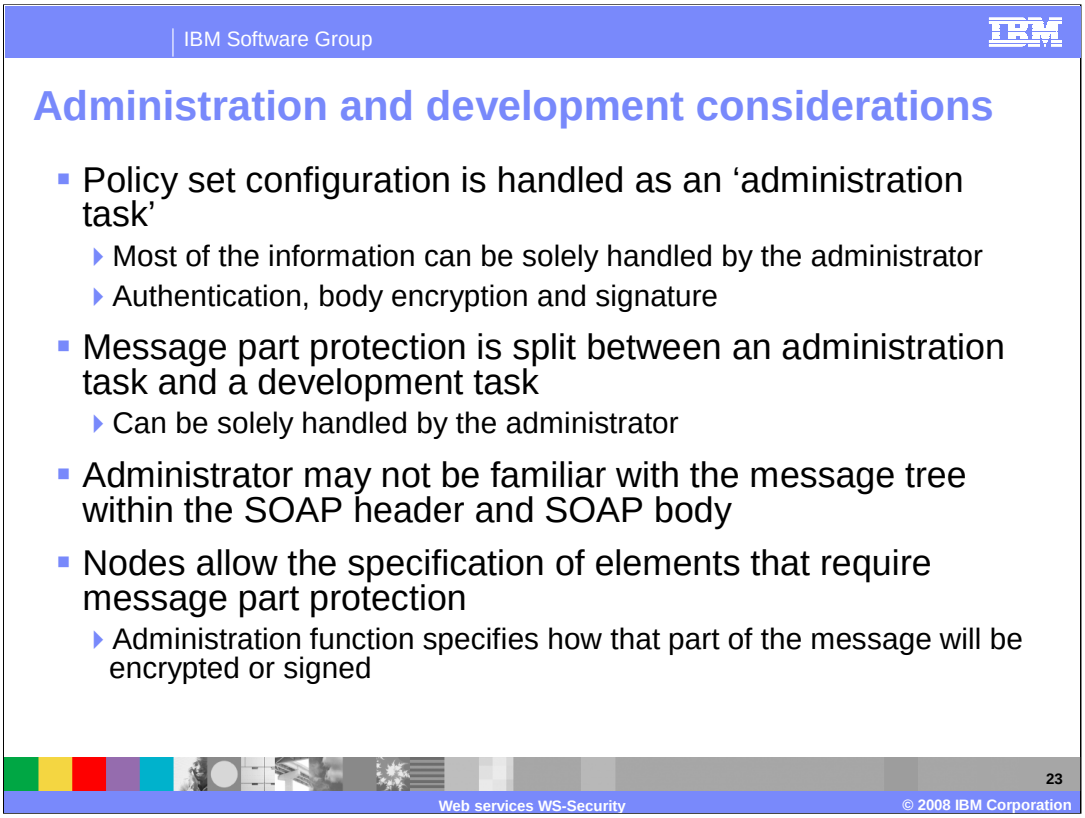
Message can be multiply encrypted and signed using different keys if required, allowing certain recipients of the message to see and change certain parts of the SOAP message but not other parts.

The policy set can also be used to encrypt header information and the body of the message. It can therefore be used in conjunction with Authentication tokens, so that if you are passing User-name and Password information in clear text, this can be encrypted and protected.

Policy set configuration overview



This slide shows an expanded screen capture from the policy set editor. It shows the key items that need to be configured, based on the particular WS-Security requirements of the application.



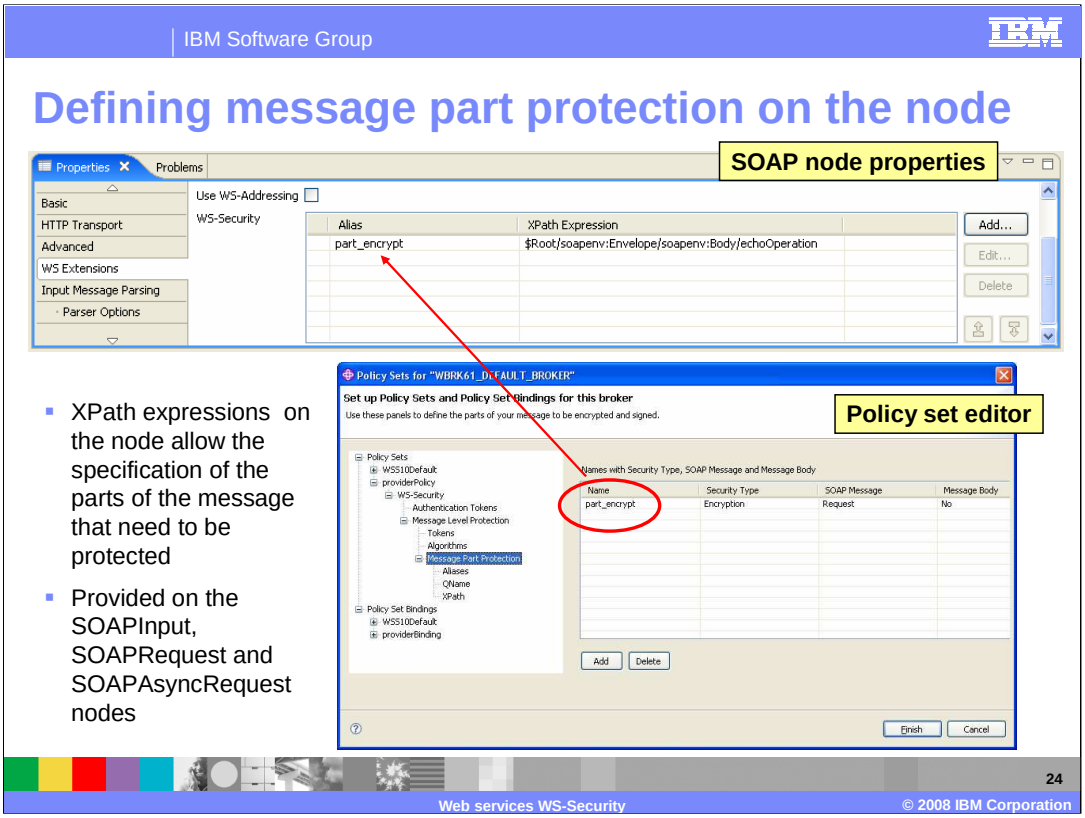
The implementation of the security requirements for a particular application are split between the application developer and the system administrator.

The majority of the implementation is done as an administration task. For Message Level protection, the administrator will be able to do all the necessary tasks to implement WS-Security.

However, administrator may not be familiar with the message tree contained within the SOAP header and SOAP body. If you need to implement Message Part Protection, you can specify which parts of the message should be protected.

The SOAP Nodes allow you to specify elements that require Message Part Protection applied.

The policy set editor allow the administrator to specify how each of these parts of the message will be encrypted or signed.



This slide shows how the properties of the SOAP Nodes are used to specify security on parts of the SOAP message.

Using the WS-Extensions tab of the SOAP node properties, an alias is added for each message part that is subject to security.

This can be the contained in the header of the message or the body of the message. X-Path expressions allow you to specify the parts of the message that need to be protected. Click “Add” on the SOAP Node properties, as shown on this screen capture. This will open a dialogue called the X-Path Expression Builder, which will allow you to specify the precise field definitions. This editor is discussed in detail in the presentation which covers the new Route Node.

The alias provides a link between the configuration provided in the SOAP nodes and the runtime components specified by the system administrator.

The Policy Set definition, performed by the system administrator, resolves the alias to either encrypt or sign the part of the message referred to by the XPath expression. This cannot be controlled SOAP node properties of the message flow.

Assigning policy sets with flows and nodes

- Different Web services may have different requirements on WS-Security
  - Different services may use different certificates
  - Different messages (for differing uses) may require different parts of the message to be encrypted
- A single policy set and policy set binding is not sufficient for all nodes in all flows
- A policy set can be assigned at the ‘flow’ or ‘node’ level depending on the required granularity

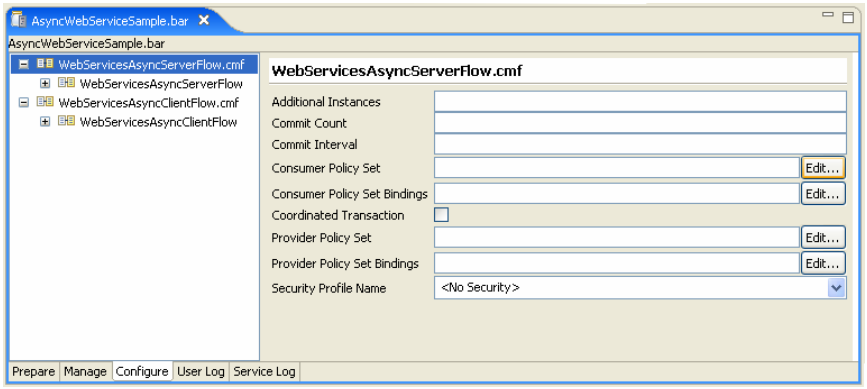
It is possible to define more than one policy set and policy set binding for each instance of a broker.

The message flows deployed to that broker may have very different requirements for Web services security. Each service may use different certificates and have different requirements for message part processing. It is therefore necessary to be able to assigning a policy set and binding to a given set of ‘Web services’, or message flows.

The policy set can be assigned at flow level, or even node level, if the requirements are very granular.

Broker archive editor

- When the top level compiled message flow entry (cmf) in the configure tree in the Broker Archive editor is selected, there are four new properties



- Text fields show the selected policy set information
- To change the policy set or binding policy set, select the Edit button
  - This will launch a dialog box that will allow selection of a previously defined Policy Set and Policy Set Binding

A policy set and binding can be assigned at the flow, or Web service level. This is done using the Bar-file editor; this is shown on the next slide.

A distinction is made between message flows which are Web service providers, and Web service consumers, or clients.

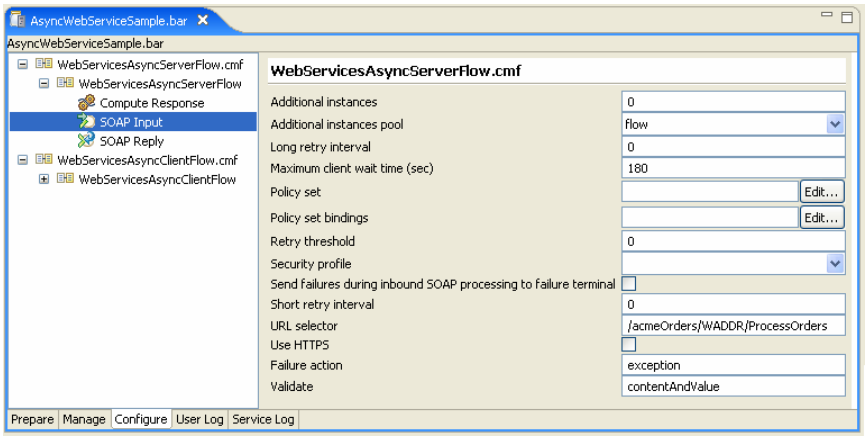
This slide shows a screen capture of the Broker Archive editor for a Web service message flow. Select the “configure” tab, then select the top-level flow-name, suffixed with “.cmf”. The fields that are required for defining the WS-Security information are the Consumer and Provider Policy Set and Policy Set Bindings.

To change the policy set or binding policy set, select the Edit button, which will launch a simple dialog window. This will query the broker for the list of defined Policy Sets or Policy Set Bindings.

Normally, you should set Policy Set first, then select the Policy Set Binding. This is because once the Policy Set has been defined, only the bindings associated with that policy are available.

Assigning a policy set at the node level

- When you select a SOAPInput, SOAPRequest or SOAPAsyncRequest entry in the Configure tree in the Broker Archive editor you will see two new properties
  - Policy Set
  - Policy Set Binding
- These new properties will work exactly the same as the flow level provider / consumer policy set property
  - Finer level of granularity



Assigning a policy set at the flow level may not be sufficiently granular, therefore, you may need to assign a Policy Set to the individual nodes.

Note that assigning a Policy Set at the flow level does not stop you from assigning a Policy Set and Binding to an individual node. The node assignment will override the assignment at the flow level, leaving any other flows to inherit the Policy Set and Binding defined at the flow level.

To do this, in the bar file editor, select the node that you want to configure, and then select the Policy Set or Policy Set Bindings as required. Specify these settings in the same way as at the flow level.

Managing policy sets using the command line

- To display a policy set and policy set binding
  - mqsireportproperties <broker> -c PolicySets -o **WSS10Default** -n ws-security
  - mqsireportproperties <broker> -c PolicySetBindings -o **WSS10Default** -n ws-security
- To introduce or create a new policy set and policy set binding
  - mqsicreateconfigurableservice <broker> -c PolicySets -o **ExamplePS**
  - mqsicreateconfigurableservice <broker> -c PolicySetBindings -o **ExamplePS**
- These can be edited (changed)
  - mqsichangeproperties <broker> -c PolicySets -o **ExamplePS** -n ws-security -p <file>
  - mqsichangeproperties <broker> -c PolicySetBindings -o **ExamplePS** -n ws-security -p <file>
- New policy set and binding files initially inherit their default values from WSS10Default policy set and binding files.

In addition to being able to define policy sets and policy set bindings within the policy set editor, you can define these using the command line.

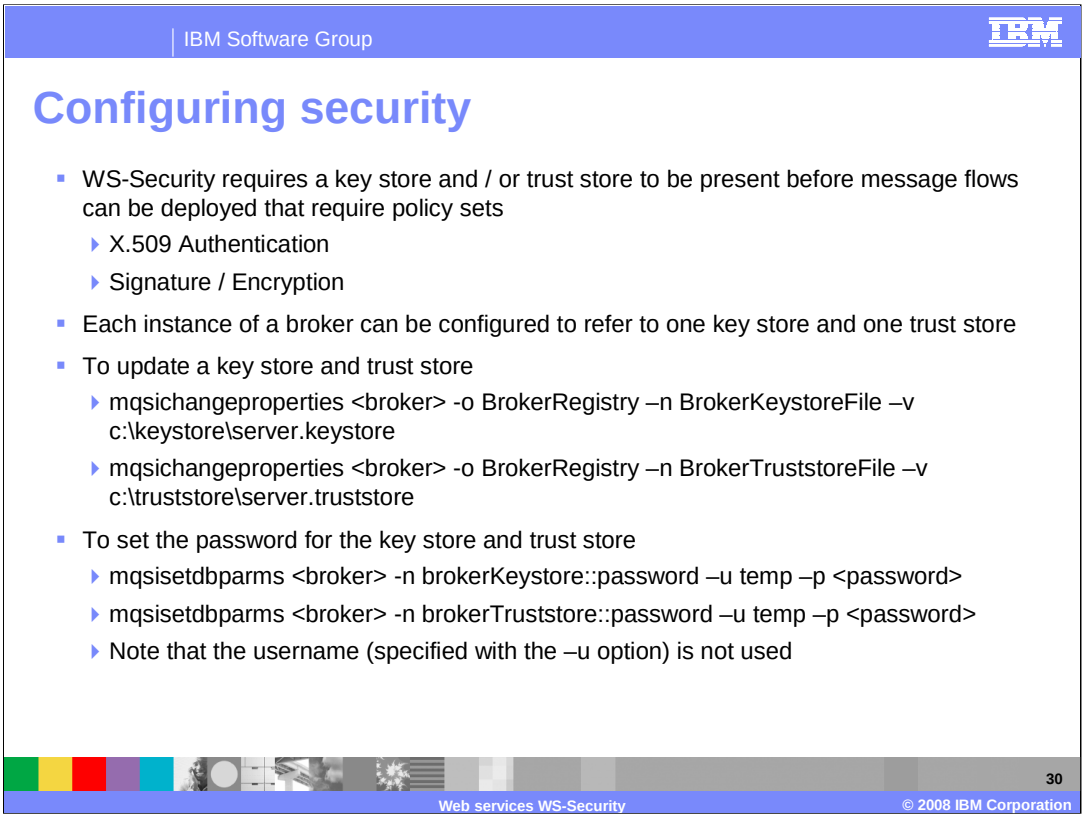
This slide shows some examples of commands used to create and maintain policy sets.

The final two commands on this slide show how the security profile can be saved to a file, or can be loaded into the broker from a saved file.

Managing policy sets using the configuration manager proxy

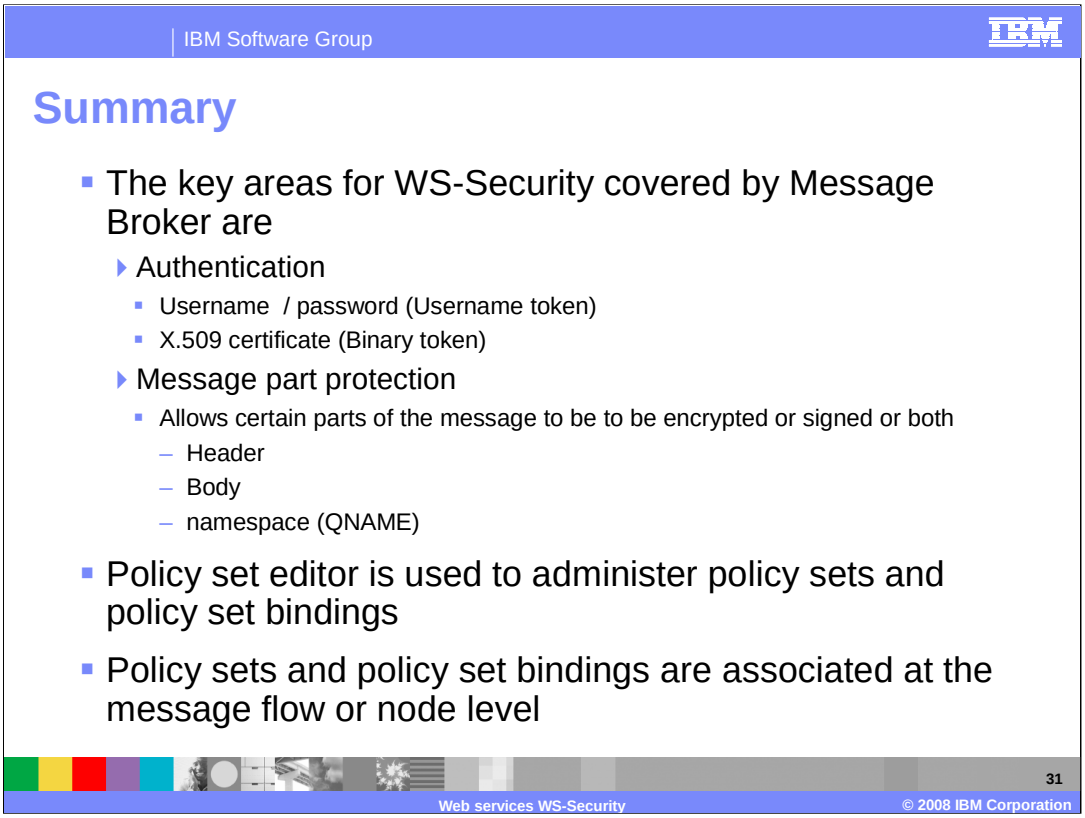
- The ***BrokerProxy*** class has been extended to include
  - public void setPolicySet(String policySetName, InputStream value)
  - public void setPolicySetBinding(String policySetBindingName, InputStream value)
  - public String getPolicySet(String policySetName)
  - public String getPolicySetBinding(String policySetBindingName)

As with most other commands, the configuration of these security profiles can also be done using the Configuration Manager Proxy API. Using this, you can write your own Java application to issue the required command. Alternatively, the Proxy API Exerciser can be used.



Finally, to enable the use of the Policy Sets and Bindings, you must also make the certificates and keys available to the Broker. This is done by adding the keys to the key store. And configuring the trust-store. The Broker is configured to use just one key store for all such security artifacts.

These commands are the same as in version 6.0 of Message Broker, since this facility was available for use with the HTTP/S nodes.



In summary, the WS-Security function in Message Broker version 6.1 covers the implementation of authentication, and the protection of SOAP messages.

Authentication can be done using user-name and password, or with certificate. This information is stored in the security header of the SOAP message.

The definition of the WS-Security function is done using the Policy Set editor; the resulting policy sets and bindings can be associated with a message flow, or a specific node within the flow.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WMB61_IEA_WebServices_WS_Security.ppt

This module is also available in PDF format at: ../WMB61_IEA_WebServices_WS_Security.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM WebSphere

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.