IBM® Tivoli® Federated Identity Manager
Version 6.2.2

Web Services Security Management Guide

IBM

IBM® Tivoli® Federated Identity Manager
Version 6.2.2

Web Services Security Management Guide

IBM

Edition notice

This edition applies to version 6, release 2, modification 2 of IBM® Tivoli®
Federated Identity Manager (product number 5724-L73) and to all subsequent
releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

ix

About this publication

IBM Tivoli Federated Identity Manager Version 6.2.2 implements solutions for federated single sign-on, Web services security management, and provisioning that are based on open standards. IBM Tivoli Federated Identity Manager extends the authentication and authorization solutions provided by IBM Tivoli Access Manager to simplify the integration of multiple existing Web solutions.

This guide describes Web services security management for Tivoli Federated Identity Manager.

Intended audience

The target audience for this book includes network security architects, system administrators, network administrators, and system integrators. Readers of this book should have working knowledge of networking security issues, encryption technology, keys, and certificates. Readers should also be familiar with the implementation of authentication and authorization policies in a distributed environment.

This book describes an implementation of a Web services solution that supports multiple Web services standards. Readers should have knowledge of specific Web services standards, as obtained from the documentation produced by the standards body for each respective standard.

Readers should be familiar with the development and deployment of applications for use in a Web services environment. This includes experience with deploying applications into an IBM WebSphere® Application Server environment.

Publications

Read the descriptions of the IBM Tivoli Federated Identity Manager library, the prerequisite publications, and the related publications to determine which publications you might find helpful. The section also describes how to access Tivoli publications online and how to order Tivoli publications.

IBM Tivoli Federated Identity Manager library

The publications in the IBM Tivoli Federated Identity Manager library are:
- *IBM Tivoli Federated Identity Manager Quick Start Guide*

  Provides instructions for getting started with IBM Tivoli Federated Identity Manager.
- *IBM Tivoli Federated Identity Manager Installation Guide*

  Provides instructions for installing IBM Tivoli Federated Identity Manager.
- *IBM Tivoli Federated Identity Manager Configuration Guide*

  Provides instructions for configuring IBM Tivoli Federated Identity Manager.
- *IBM Tivoli Federated Identity Manager Administration Guide*

  Provides instructions for completing administration tasks that are required for all deployments.
- *IBM Tivoli Federated Identity Manager Web Services Security Management Guide*

Provides instructions for completing configuration tasks for Web services security management.

- *IBM Tivoli Federated Identity Manager Auditing Guide*

  Provides instructions for auditing IBM Tivoli Federated Identity Manager events.
- *IBM Tivoli Federated Identity Manager Error Message Reference*

  Provides explanations of the IBM Tivoli Federated Identity Manager error messages.
- *IBM Tivoli Federated Identity Manager Troubleshooting Guide*

  Provides troubleshooting information and instructions for problem solving.

You can obtain the publications from the IBM Tivoli Federated Identity Manager Information Center:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/ com.ibm.tivoli.fim.doc_6.2.2/ic/ic-homepage.html

Prerequisite publications

To use the information in this book effectively, you should have some knowledge about related software products, which you can obtain from the following sources:

- Tivoli Access Manager Information Center:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?toc=/ com.ibm.itame.doc/toc.xml
- IBM WebSphere Application Server Version 8.0 Information Center:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

  You can obtain PDF versions of the IBM WebSphere Application Server documentation at:

  http://www.ibm.com/software/webservers/appserv/was/library/

Related publications

You can obtain related publications from the IBM Web sites:

- *Enterprise Security Architecture Using IBM Tivoli Security Solutions*. This book is available in PDF (Portable Document Format) at http:// www.redbooks.ibm.com/redbooks/pdfs/sg246014.pdf or in HTML (Hypertext Markup Language) at http://www.redbooks.ibm.com/redbooks/SG246014/
- *Federated Identity Management and Web Services Security with IBM Tivoli Security Solutions* (SG24-6394-01). This book is available in PDF at http:// www.redbooks.ibm.com/redbooks/pdfs/sg246394.pdf or in HTML at http://www.redbooks.ibm.com/redbooks/SG246394/
- The Tivoli Software Library provides a variety of Tivoli publications such as white papers, datasheets, demonstrations, redbooks, and announcement letters. The Tivoli Software Library is available on the Web at: http:// publib.boulder.ibm.com/tividd/td/tdprodlist.html
- The *Tivoli Software Glossary* includes definitions for many of the technical terms related to Tivoli software. The *Tivoli Software Glossary* is available at http://publib.boulder.ibm.com/tividd/td/tdprodlist.html

Accessing terminology online

The IBM Terminology Web site consolidates the terminology from IBM product libraries in one convenient location. You can access the Terminology Web site at http://www.ibm.com/software/globalization/terminology

Accessing publications online

IBM posts publications for this and all other Tivoli products, as they become available and whenever they are updated, to the Tivoli Information Center Web site at http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp.

Note: If you print PDF documents on other than letter-sized paper, set the option in the **File ⁊ Print** window that allows Adobe Reader to print letter-sized pages on your local paper.

Ordering publications

You can order hard copies of some publications.

Many countries provide an online ordering service.
Follow these steps to access this service:

1. Go to http://www-947.ibm.com/support/entry/portal/Documentation
2. Select **IBM Publications Center** from **Getting Started**.
3. Select your country from **Select a country/region/language to begin** and click the arrow icon.
4. Follow the instructions for how to order hard copy publications on Welcome to the IBM Publications Center.

If your country does not provide an online ordering service, contact your software account representative to order publications.
Follow these steps to find your local contact:

1. Go to http://www.ibm.com/planetwide/
2. Click your country name to display a list of contacts.

Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. With this product, you can use assistive technologies to hear and navigate the interface. You also can use the keyboard instead of the mouse to operate all features of the graphical user interface.

For additional information, see the "Accessibility" topic in the information center at http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.tivoli.fim.doc_6.2.2/ic/ic-homepage.html.

Tivoli technical training

For Tivoli technical training information, refer to the following IBM Tivoli Education Web site at http://www.ibm.com/software/tivoli/education.

Support information

If you have a problem with your IBM software, you want to resolve it quickly. IBM provides the following ways for you to obtain the support you need:

Online
Go to the IBM Software Support site at http://www.ibm.com/software/support/probsub.html and follow the instructions.

IBM Support Assistant
The IBM Support Assistant (ISA) is a free local software serviceability

workbench that helps you resolve questions and problems with IBM software products. The ISA provides quick access to support-related information and serviceability tools for problem determination. To install the ISA software, see the *IBM Tivoli Federated Identity Manager Installation Guide*. Also see: http://www.ibm.com/software/support/isa.

Troubleshooting Guide
> For more information about resolving problems, see the *IBM Tivoli Federated Identity Manager Troubleshooting Guide*.

Conventions used in this book

This reference uses several conventions for special terms and actions and for operating system-dependent commands and paths.

Typeface conventions

This publication uses the following typeface conventions:

Bold
- Lowercase commands and mixed case commands that are otherwise difficult to distinguish from surrounding text
- Interface controls (check boxes, push buttons, radio buttons, spin buttons, fields, folders, icons, list boxes, items inside list boxes, multicolumn lists, containers, menu choices, menu names, tabs, property sheets), labels (such as **Tip:**, and **Operating system considerations**:)
- Keywords and parameters in text

Italic
- Citations (examples: titles of publications, diskettes, and CDs
- Words defined in text (example: a nonswitched line is called a *point-to-point line*)
- Emphasis of words and letters (words as words example: "Use the word *that* to introduce a restrictive clause."; letters as letters example: "The LUN address must start with the letter *L*.")
- New terms in text (except in a definition list): a *view* is a frame in a workspace that contains data.
- Variables and values you must provide: ... where *myname* represents....

Monospace
- Examples and code examples
- File names, programming keywords, and other elements that are difficult to distinguish from surrounding text
- Message text and prompts addressed to the user
- Text that the user must type
- Values for arguments or command options

Operating system-dependent variables and paths

This publication uses the UNIX convention for specifying environment variables and for directory notation.

When using the Windows command line, replace $*variable* with % *variable*% for environment variables and replace each forward slash (/) with a backslash (\) in directory paths. The names of environment variables are not always the same in

the Windows and UNIX environments. For example, %TEMP% in Windows environments is equivalent to $TMPDIR in UNIX environments.

Note: If you are using the bash shell on a Windows system, you can use the UNIX conventions.

Chapter 1. Overview of Web services security management

The Web services security management component of IBM Tivoli Federated Identity Manager is used to establish and manage federation relationships for WebSphere Web service applications that use WS-Security tokens.

The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

The Web services security management component provides functions for:
- Web service providers, that use a token consumer to process inbound security tokens
- Web service clients, that use a token generator to create outbound security tokens

Token processing and creation are done through a series of Web service request and response messages that interact with the Web services security management component, and the Tivoli Federated Identity Manager trust service.

Note: IBM deprecates the Tivoli Federated Identity Manager Security Token Service (STS) Client in this release.

If you use WebSphere 6.X, you can still use the Tivoli Federated Identity Manager Security STS client while Tivoli Federated Identity Manager supports WebSphere 6.X. When Tivoli Federated Identity Manager discontinues its support for WebSphere 6.X, use WebSphere Application Server version 7 Update 11 and later. See WS-Trust client API and WS-Trust Clients for details.

The Web services security management component also provides Web services applications with an authorization solution and identity and security token mapping capabilities that can be used without deploying a federated single sign-on environment. Unlike the other Tivoli Federated Identity Manager components, the Web services security management component does not require the use of Tivoli Access Manager WebSEAL component.

The Web services security management component enhances the WS-Security support provided by WebSphere Application Server in a number of ways:
- Extends the WS-Security token types available in WebSphere Application Server to additional token types supported by the Tivoli Federated Identity Manager trust service. For example, in this feature, you can directly use a Security Assertion Markup Language (SAML) assertion for authentication.
- You can exchange the type of token; for example, a UsernameToken can be exchanged for a SAML assertion.
- You can map the user identity; for example, a user identity can be mapped to a local identity expected by the Web service.
- You can map the user identity to many-to-one and one-to-many.

- You can do authorization checks on a Web service before starting the Web service. Using the capabilities provided by Tivoli Access Manager for e-business, requests can be validated without running the underlying Web service.

All of these features are available at both the Web service client and the Web service.

Web services security management component and product interactions

The Web services security management component interacts with Tivoli Federated Identity Manager components and other products.

Figure 1 on page 3 illustrates the product and component interactions. The description of each labeled interactions, denoted by a letter from A to E or a number from 1 to 6, is described following the diagram.
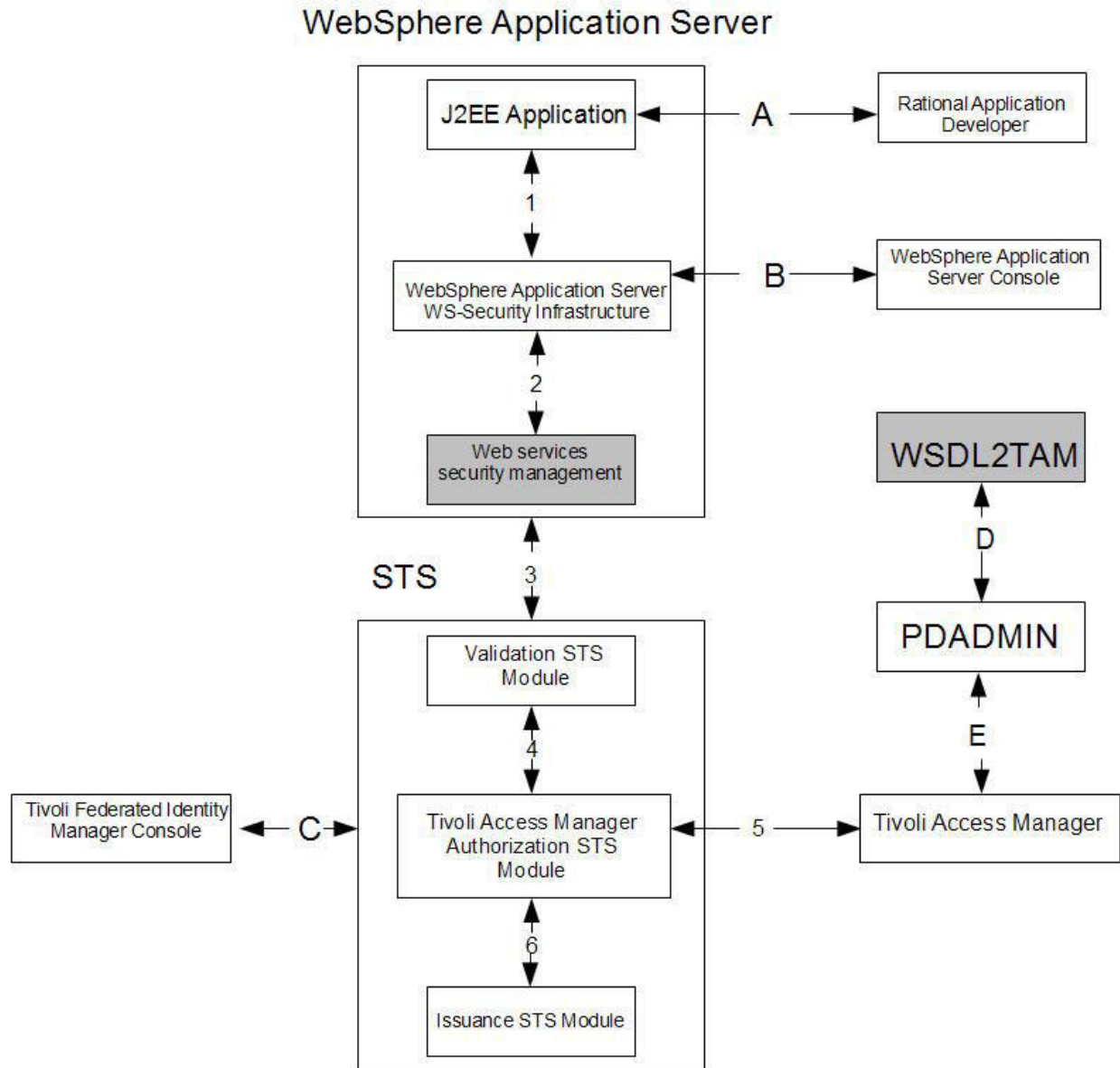
Figure 1. Web services security management components

In Figure 1, the following letters summarize the deployment and configuration interactions between the components and products.

A. A J2EE application, acting as either or both a Web service consumer or provider, is configured to use Web services security management for token generation, token consumption, or both. The Rational® Application Developer is used to update the WS-Security deployment descriptors with Web services security management configuration. The application does not run Web services security management directly.

B. Optionally, you can use the WebSphere Application Server console to modify the WS-Security configuration bindings of the J2EE application.

C. You can use the Tivoli Federated Identity Manager console to configure the Web services security management runtime to start the STS module chain.

D. Optionally, you can use the wsdl2tam command-line utility to create a script for creating the Tivoli Access Manager protected objects representing the Web

service. This step is only required if the Tivoli Access Manager authorization STS module is included in the STS module chain.

E. The script generated by WSDL2TAM is run by PDADMIN to update the Tivoli Access Manager protected object space.

In Figure 1 on page 3, the following numbers summarize the runtime interactions between the components and products.

1. WebSphere Application Server starts the WS-Security infrastructure on behalf of the J2EE application. This process can occur in two possible ways. One way is if the application is about to use a Web service. In this case, a security token must be generated. Another way is if a Web service provided by the application is being called. In this case, a security token must be used.

2. The WS-Security infrastructure delegates the security token processing to the Web services security management token generator or consumer, which uses the STS.

3. Web services security management sends a RequestSecurityToken message to the STS including a security token, issuer, and AppliesTo. The STS runs the module chain configured for the security token, issuer, and AppliesTo. For brevity, the optional mapping STS modules are not shown in the figure.

4. After token validation, the Tivoli Access Manager authorization STS module is called. Inclusion of this module in the chain is optional. Authorization might occur as part of token generation or, more typically, token consumption.

5. The Tivoli Access Manager authorization STS module makes an authorization check against Tivoli Access Manager to ensure that the user is authorized to start the Web service operation.

6. After passing the authorization check, the token issuing the STS module is called. This token is returned to the Web services security management token generator or consumer. The Web services security management token generator includes the token in the WS-Security header. The Web services security management token consumer uses the token to log in.

Preparing Web services security management for your Web service application

Before you use Web services security management component with your Web service application, you must perform some installation and configuration tasks.

1. Install the Web services security management component, as described in the *IBM Tivoli Federated Identity Manager Installation Guide*.

2. Perform some configuration tasks for the component and its required software, including:

   a. Chapter 2, "Configuring WebSphere Application Server," on page 27.

   b. Chapter 3, "Updating configuration batch and script files," on page 39.

   c. Chapter 4, "Configuring Tivoli Access Manager," on page 45, if you plan to use the authorization function of the Web services security management component.

   d. Chapter 5, "Configuring keystores and certificates," on page 49, if keys or certificates are used with the tokens you plan to validate or exchange.

Preparing your Web service application to use Web services security management

This guide provides the steps required to create a connection between a previously created and configured Web service application and the Web services security management component.

This guide does not provide the steps required to create a WebSphere Web services application that uses WS-Security standards. See the Rational Application Developer documentation, which contains information about developing Web services applications.

Follow these steps to use Web service application with the Web services security management component:

1. Configure your Web service application to use the Web services security management component.

   This process is described in the following areas of this guide:

   - "Component summary" on page 7 describes the component parts, their class names, and parameters.
   - Chapter 6, "Configuring your Web services application to use Web services security management," on page 51 describes the deployment descriptors of an application necessary to generate and use tokens.
   - "Echo demonstration application" on page 18 shows a sample application that is provided with the component which you can use as a guide for configuring your application.

2. Configure the Tivoli Federated Identity Manager trust service to be used with your Web service application.

   You must create STS module chains using the Tivoli Federated Identity Manager administrative console, to validate and issue tokens for WSSM. This process is described in the *IBM Tivoli Federated Identity Manager Administration Guide*. This book also provides general information about the trust service, modules, module instances, and module chains.

Supported token module types

The Web services security management component supports a set of token module types by default.

AuthorizationSTSModule
> If you currently use this module type, upgrade to TAMAuthorizationSTSModule.

DynamicChainSelectionModule
> Requests that the trust service calls a dynamic chain after processing is finished on the current module chain. This module is used by Web services security management. This type is not used in federated single sign-on.

DirectoryIntegratorSTSModule
> Performs generic user and attribute mapping functions. An assembly line running on a Tivoli Directory Integrator server is called to map the user and attribute data in an STSUniversalUser. Data might be resolved from various data sources natively supported by Tivoli Directory Integrator, including LDAP and relational databases. Custom code is also supported through JavaScript connectors.

KerberosSTSModule

Provides support for the use of Kerberos tokens. The following Kerberos V5 (WS-Security Kerberos Token Profile) security tokens are supported:

- GSS-wrapped Kerberos V5 AP-REQ
- Non-wrapped Kerberos V5 AP-REQ

PassTicketSTSModule

Provides support for creating or the use of PassTicket tokens. In a Resource Access Control Facility (RACF®) secured signon, a PassTicket is a dynamically generated, random, one-time-use, password substitute that a workstation or other client can use to sign on to the host, instead of sending a RACF password across the network.

This module can generate PassTickets on non-z/OS environments if the secret key is shared.

Note: The Web services security management component supports an RACF PassTicket token even if it is not a token type defined by the OASIS WS-Security standards. You can also use the RACF PassTicket token like an OASIS Username token in the configurations depicted in this guide.

SAML10STSModule (SAML 1.0)

Provides support for creating or the use of SAML 1.0 tokens.

SAML20STSModule (SAML 2.0)

Provides support for creating or the use of SAML 2.0 tokens.

SAMLTokenSTSModule

Provides support for creating or the use of SAML tokens.

STSLTPATokenModule

Provides support for creating or the use of LTPA V1 and LTPA V2 tokens. These tokens are represented as BinarySecurityToken elements.

STSMapDefault

Provides support for Extensible Stylesheet Language Transformation (XSLT) mapping files. All federation configurations use this module as the default mapping module.

STSMessageLogger

Tracks all, or part of a Security Token Service transaction, including requests, responses, incoming and outgoing mappings, errors and exceptions.

STSTokenIVCred

Provides support for creating or the use of Tivoli Access Manager credentials.

TAMAuthenticationSTSModule

Provides support for a Tivoli Access Manager authentication with the use of the supplied credentials.

TAMAuthorizationSTSModule

Provides support for a Tivoli Access Manager authorization check using the supplied user identity and resource name.

UsernameTokenSTSModule (Username Token)

Provides support for creating or using tokens in a WebSphere Web services environment where the Username token is used to pass user identity information in the header of a Web services request.

X509STSModule

Provides support for the use of X.509 tokens by performing certificate path validation on certificates, and certificate paths contained within the tokens.

The following module types, which are supported by Tivoli Federated Identity Manager in single sign-on federations, are not supported by the Web services security management component:

- DelegatorSTSModule
- DSigSTSModule
- JAASSTSModule
- Liberty12SAMLTokenSTSModule
- LibertySAMLTokenSTSModule
- STSMessageLoggerModule
- STSTAMGSOModule

Component summary

The Web services security management component consists of many parts.

These parts are:
- Token generator
- Callback handlers
- Token consumers
- Login modules

The token generator and callback handlers are used by Web service clients requesting a security token to be inserted into the Web service request.

Web services also use the token consumer and login modules to do the following:
- Process the security token received in a Web service request
- Create a login context to process the Web service request

The token consumer, token generator, callback handlers, and login modules comply with the WebSphere Application Server WS-security model for adding support for additional token types. See the Web services information that is part of the WebSphere Application Server Information Center for details.

WSSM Token generator

The WSSM token generator module is a WebSphere Application Server token generator that delegates creation of a security token to a configured callback handler. The token generator can use the security token obtained to optionally call the Tivoli Federated Identity Manager trust service to do token validation, token exchange, and an authorization check.

The token generator module is called as part of the WebSphere Application Server WS-Security authentication processing when a request message is created. The token generator delegates token generation to the WSSM token generator callback handler. If the Web service client application uses this token generator but is not configured to call the trust service, the token generated by the callback handler is placed in the security header of the request message. The Web service client application is not configured to call the trust service when the sts.call value is set to false.

However, if the Web service client uses this token generator and is configured to call the trust service, then the token generator creates a RequestSecurityToken message. The RequestSecurityToken message is then sent to the trust service. The token generator does the following tasks:

- Sets the `Issuer` to `urn:itfim:wssm:tokengenerator`
- Sets the `AppliesTo` to the URL of the Web service application
- Includes the generated security token in the message

The trust service processes the request and returns a RequestSecurityTokenResponse message to the WSSM token generator. The generator then includes the token from the trust service response message in the security header of the Web service request message.

If no token is returned or an error occurs, then the token generation fails. The Web service is not called, and an error is returned to the Web services client.

You can use the token generator module and trust service to do the following actions:

- The exchange of security tokens when the incoming or outgoing security tokens are of different types
- The exchange of security tokens when mapping one identity to another
- The evaluation of authorization checks to ensure that authenticated users can start the target Web service

The WSSM token generator class name is `com.tivoli.am.fim.wssm.tokengenerators.WSSMTokenGenerator`. This class name must be specified in the Web services security binding configuration to have the WSSM token generator started.

Supported token types

The token generator supports the exchange or evaluation of the following security token types:

- SAML 1.0
- SAML 1.1
- SAML 2.0
- Kerberos
- Username
- PassTicket
- Tivoli Access Manager

Note: The only support provided for X.509 tokens is for consuming them. Therefore, they are not supported by the token generator.

The token generator always passes the credentials of the Web service client to the trust service in a SAML 1.1 or SAML 2.0 token.

Parameters

The following token generator module properties can be configured in your Web service application.

Table 1. Token generator module configuration parameters

Parameter name	Description	Default value
issuer	The issuer address to use in the request security token message to the STS. Normally, the default issuer is sufficient. One scenario where you might want to specify an issuer would be when using the token generator and two different client applications accessing the same Web service. But, in that scenario, you want the two clients to use different token types. By specifying different issuer values in the deployment descriptor for each client application, you can define different STS module chains. Then you can generate different security tokens to send to the Web service.	`urn:itfim:wssm:tokengenerator`
`sts.call`	A Boolean flag indicating whether the Tivoli Federated Identity Manager trust service is to be called. If the trust service is not called, the original security token is used for authentication. Valid values: `true`, `false`.	`true`

Table 1. Token generator module configuration parameters (continued)

Parameter name	Description	Default value
`unique.token.id.xpath`	The XPath that is evaluated against the security token to generate a unique identifier. The unique identifier uniquely identifies the security token to WebSphere Application Server. For some token types, the default unique identifier is appropriate because the information used to create the identifier is always unique. However, in cases where the identifier might be the same for multiple tokens when in fact they must be unique, or in which the identifier is unique for multiple tokens when they must be the same, you must use `unique.token.id.xpath`. For example, for SAML assertions, the default unique identifier is the SAML assertion identifier. Multiple SAML assertions, each with the same authentication and attribute statements, have different identifiers. However, you might want to use `unique.token.id.xpath` so that the identifier is the same. WebSphere Application Server keeps a cache of login credentials. If an identifier matches the cached credentials, the performance of the WebSphere Application Server login processing can be improved.	`null` With a value of `null`, a unique identifier is determined based on the token type; for example, the SAML assertion identifier is used for SAML assertions.

Callback handlers

The callback handler modules perform specialized processing on Web service messages or tokens, as needed.

There are several types of callback handler modules that are part of the Web services security management component.

WSSM token generator callback handler and delegate handlers

The WSSM token generator callback handler is used by the WSSM token generator to create XML and WebSphere Application Server security tokens.

When the WSSM token generator is started, it calls the WSSM token generator callback handler to create an XML security token. The WSSM token generator callback handler delegates this creation to the configured xml.callback.handler.class.name handler. The WSSM token generator then optionally calls the trust service, passes it this XML security token, and receives a returned XML security token. If this call has been configured to not occur, then the original token is used. The WSSM token generator then calls the WSSM token

generator callback handler to create a WebSphere Application Server token object from the XML security token. The WSSM token generator callback handler delegates this creation to the configured token.callback.handler.class.name handler. When this process is done, WebSphere Application Server has enough information to insert the security token in the outgoing Web service request.

The WSSM token generator callback handler class name is com.tivoli.am.fim.wssm.callbackhandlers.WSSMTokenGeneratorCallbackHandler.

The callback handlers for generating XML and WebSphere Application Server security tokens are as follows:

- XML callback handlers:

  JAAS subject credential callback handler
  > This callback handler returns the JAAS subject in a SAML 1.1 or SAML 2.0 assertion, configured by the property saml.version. The property saml.version can be either 1.1 or 2.0, and defaults to 2.0. The principal name of the subject is contained within a SAML assertion authentication statement. The private and public credentials from the subject are included as attributes in the attribute statement of the SAML assertion.

  WSSM credential callback handler
  > This callback handler returns the XML security tokens contained within the WSSM private credential of the JAAS Subject.

- WebSphere Application Server callback handlers:

  Kerberos callback handler
  > Use the Kerberos callback handler to generate Kerberos security tokens.

  SAMLA callback handler
  > Use the SAMLA callback handler to generate SAML security tokens.

  Username callback handler
  > Use the Username callback handler to generate Username security tokens.

  Tivoli Access Manager callback handler
  > Use the Tivoli Access Manager callback handler to generate Tivoli Access Manager security tokens.

Parameters

The following WSSM token generator callback handler properties can be configured. The delegate callback handlers do not have configurable parameters.

Table 2. WSSM token generator callback handler configuration parameters

Parameter name	Description	Default value
xml.callback.handler.class.name	The class name of the callback handler responsible for creating XML security tokens. Valid values: com.tivoli.am.fim.wssm.callbackhandlers. JAASSubjectCallbackHandler com.tivoli.am.fim.wssm.callbackhandlers. WSSMCredentialCallbackHandler	None.

Table 2. WSSM token generator callback handler configuration parameters (continued)

Parameter name	Description	Default value

`token.callback.handler.class.name`	The class name of the callback handler responsible for creating WebSphere Application Server security tokens.
	
	Valid values:
	
	`com.tivoli.am.fim.wssm.callbackhandlers.`
	`KerberosCallbackHandler`
	
	`com.tivoli.am.fim.wssm.callbackhandlers.`
	`SAMLACallbackHandler`
	
	`com.tivoli.am.fim.wssm.callbackhandlers.`
	`UsernameCallbackHandler`
	
	`com.tivoli.am.fim.wssm.callbackhandlers.`
	`TAMCredentialCallbackHandler`
		None.

WSSM token consumer callback handler

The WSSM token consumer callback handler is used by the WSSM token consumer to pass a security token to a configured login module.

The WSSM token consumer is called as part of the WebSphere Application Server WS-Security authentication processing when a Web service request message is received.

If the trust service is not used to fulfill a request, the token is passed from the WSSM token consumer to the callback handler. The callback handler then passes it to the appropriate login module.

If the trust service is called by the token consumer to fulfill a request, the token consumer creates a RequestSecurityToken message. Then, it sends the message to the trust service. The trust service processes the request and returns a RequestSecurityTokenResponse message to the token consumer. The token returned from the trust service is then passed to the callback handler. The callback handler makes the token available to the appropriate login module and the token is used to log in.

The WSSM token consumer callback handler class name is `com.tivoli.am.fim.wssm.callbackhandlers.WSSMTokenConsumerCallbackHandler`.

Parameters

There are no configuration parameters for this callback handler.

Token consumers

The token consumer modules are WebSphere Application Server token consumers that use the Tivoli Federated Identity Managertrust service to do token validation, token exchange, and an authorization check.

There are two token consumers in the Web services security management component:
- **WSSM token consumer.** The WSSM token consumer receives a token, calls the trust service, unless the `sts.call` that is configured for this consumer in your Web services application is `false`. Then, the WSSM token consumer passes the returned token to the configured login module.

- **SAMLA token consumer.** The SAMLA token consumer receives a SAML assertion and passes it to the SAMLA login module.

WSSM token consumer

The WSSM token consumer receives tokens and calls the trust service to validate, exchange, and optionally check their authorization.

The WSSM token consumer is called as part of the WebSphere Application Server WS-Security authentication processing when a Web service request message is received.

If the Web service application uses this token consumer, but is not configured to call the trust service, the input token in the message is used to log in. The Web service application is not configured to call the trust service when `sts.call` is set to `false`.

However, if the Web service application uses this token consumer, and is configured to call the trust service, then the token consumer creates a RequestSecurityToken message. The RequestSecurityToken message is then sent to the trust service. The token consumer sets the `Issuer` to `urn:itfim:wssm:tokenconsumer`, sets the `AppliesTo` value to the URL of the Web service application, and includes the received security token in the message. The trust service processes the request and returns a RequestSecurityTokenResponse message to the WSSM token consumer. The token returned by the trust service is the one used to perform a login. The target Web service starts.

If no token is returned, or an error occurs, then the login attempt fails. The Web service is not called, and an error is returned to the Web service client.

You can use the token consumer module and the trust service to do the following actions:
- The validation of incoming security tokens
- The exchange of security tokens when the incoming or outgoing security tokens are of different types
- The exchange of security tokens when mapping one identity to another
- The evaluation of authorization checks to ensure that authenticated users can start the target Web service

The WSSM token consumer class name is `com.tivoli.am.fim.wssm.tokenconsumers.WSSMTokenConsumer`. This class name must be specified in the Web services security binding configuration to have the WSSM token consumer started.

Supported token types

The WSSM token consumer can receive the following security token types in a message from a partner Web service client:
- SAML 1.0
- SAML 1.1
- SAML 2.0
- Username
- Kerberos
- PassTicket

- X.509
- Tivoli Access Manager

The WSSM token consumer can receive the following security token types that can be issued by the trust service:
- SAML 1.0
- SAML 1.1
- SAML 2.0
- Username
- PassTicket
- Tivoli Access Manager

Parameters

The following WSSM token consumer module properties can be configured in your Web service application.

Table 3. WSSM token consumer module configuration parameters

Parameter name	Description	Default value
issuer	The issuer address to use in the request security token message to the STS.	`urn:itfim:wssm:tokenconsumer`
`sts.call`	A Boolean flag indicating whether the Tivoli Federated Identity Manager trust service is to be called. If the trust service is not called, the original security token is used for authentication. Valid values: `true`, `false`.	`true`

Table 3. WSSM token consumer module configuration parameters (continued)

Parameter name	Description	Default value
`unique.token.id.xpath`	The XPath that is evaluated against the security token to generate a unique identifier. The unique identifier uniquely identifies the security token to WebSphere Application Server. For some token types, the default unique identifier is appropriate because the information used to create the identifier is always unique. However, in cases where the identifier might be the same for multiple tokens when in fact, they must be unique, or in which the identifier is unique for multiple tokens when they must be the same, you must use `unique.token.id.xpath`. For example, for SAML assertions, the default unique identifier is the SAML assertion identifier. Multiple SAML assertions, each with the same authentication and attribute statements, have different identifiers. However, you might want to use `unique.token.id.xpath` so that the identifier is the same. WebSphere Application Server keeps a cache of login credentials. If an identifier matches the cached credentials, the performance of the WebSphere Application Server login processing can be improved.	`null` With a value of `null`, a unique identifier is determined based on the token type; for example, the SAML assertion identifier is used for SAML assertions.

SAMLA token consumer
The SAMLA token consumer uses only SAML assertion security tokens that are in the Web service requests.

If the SAMLA token consumer is configured as the token consumer in the WS-Security deployment descriptor of the Web service application, the request containing the SAML assertion is sent to the SAMLA token consumer module for processing. The SAMLA token consumer sends the request to the SAMLA login module for further processing.

The SAMLA token consumer name is `com.tivoli.am.fim.wssm.tokenconsumers.SAMLATokenConsumer`.

Supported token types

The SAMLA token consumer can receive the following security token types issued by the trust service:
- SAML 1.0

- SAML 1.1
- SAML 2.0

Parameters

There are no configuration parameters for the SAMLA token consumer.

Login modules

The login modules are Java™ Authentication and Authorization Service (JAAS) login modules that use credentials contained within security tokens to perform a login.

The login modules are called by the WSSM token consumer as part of the WebSphere Application Server WS-Security authentication processing when a request message is received. There are three login modules in the Web services security management component:
- SAMLA login module. The SAMLA login module uses the credentials in the SAML assertion security token to log in.
- Username login module. The Username login module uses the credentials in the Username security token to log in.
- Tivoli Access Manager login module. The Tivoli Access Manager login module uses the credentials in the Tivoli Access Manager token to log in.

SAML assertion login module

The SAML assertion login module is a Java Authentication and Authorization Service (JAAS) login module that accepts SAML assertions to log in.

Note: The SAML assertion login module does not use the Tivoli Federated Identity Manager trust service.

The SAML assertions are not validated by the SAML assertion login module because the SAML assertion is assumed to have come from a trusted source that has already validated it. An example of a trusted source that has already validated is the trust service. Therefore, conditions are not checked, and any digital signatures that are present are not verified by the login module. The following are the requirements:
- SAML 1.0 and 1.1 assertions contain an authentication statement that includes a subject with a name identifier
- SAML 2.0 assertions contain a subject with a name identifier

The SAML assertion login module uses the login context of the principal created from the SAML assertion to run the target Web service. This process also makes the attribute statements associated with the SAML assertion available to the target Web service through private credentials.

The SAML assertion login module class name is `com.tivoli.am.fim.wssm.loginmodules.SAMLALoginModule`.

The preferred JAAS configuration name is `system.itfim.wssm.samla`.

Parameters

There are no configuration parameters for the SAML assertion login module.

Username login module

The Username login module is a Java Authentication and Authorization Service (JAAS) login module that accepts Username security tokens to perform a login.

Note: The Username login module does not use the trust service of Tivoli Federated Identity Manager.

The Username tokens are not validated by the Username login module because the Username token is assumed to have come from a trusted source. A trusted source, such as the trust service, validates Username tokens before it goes to the Username login module. Therefore, any password that is present is not verified by the login module. The only requirement is that the Username token includes a user name.

The Username login module uses the login context of the principal created from the Username token, to run the target Web service.

The Username login module class name is
`com.tivoli.am.fim.wssm.loginmodules.UsernameLoginModule`.

The preferred JAAS configuration name is `system.itfim.wssm.username`.

Note: The WebSphere Application Server supplied JAAS configurations, such as wssecurity.UsernameToken, cannot be used with the Web services security management component because the credentials are presented in a different manner.

Parameters

There are no configuration parameters for the Username login module.

Tivoli Access Manager login module

The Tivoli Access Manager login module is a Java Authentication and Authorization Service (JAAS) login module that accepts Tivoli Access Manager security tokens to perform a login. The Tivoli Access Manager login module does not use the Tivoli Federated Identity Manager trust service.

The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

The Tivoli Access Manager login module permits login using a binary security token containing a privileged attribute certificate (PAC), which is known as a Tivoli Access Manager token. A PDPrincipal is created from the PAC and the principal name is used for login. The login module does not look up the principal name in a user registry.

The Tivoli Access Manager login module uses the login context of the principal created from the Tivoli Access Manager token to run the target Web service.

The Tivoli Access Manager login module class name is
`com.tivoli.am.fim.wssm.loginmodules.TAMLoginModule`.

The preferred JAAS configuration name is `system.itfim.wssm.tam`.

To generate and use Tivoli Access Manager security tokens, PDJRTE must be configured and the PDJRTE configuration file specified to WSSM. This PDJRTE configuration file is used to create the PDPrincipal from the privileged attribute certificate (PAC). The WSSM configuration is specified in the wssm.properties file located at `C:\Program Files\IBM\FIM\wssm\etc\wssm.properties`. Update the pdjrte.configuration property and specify the PDJRTE configuration file path.

Parameters

There are no configuration parameters for the Tivoli Access Manager login module.

JAAS login security token

The Web services security management component makes the following credentials available to the Web service application: the security token received by the Web service, and the security token that is used for JAAS login.

If the trust service is called, then the security token received by the Web service is the token that is sent to the trust service for validation. The token returned by the trust service is used to perform a JAAS login. If the trust service is not called, then the received security token is used as the JAAS login security token.

These tokens are accessed through a com.tivoli.am.fim.wssm.credentials.WSSMCredential object that is a private credential in the JAAS Subject object. The WSSMCredential object includes a `getReceivedSecurityToken` method that returns the security token, as a DOM element, that was received by the Web service. It also includes a `getLoginSecurityToken` method that returns the security token, as a DOM element, that was used to perform a JAAS login.

Access to these tokens from the Web service application might be useful. For example, if the JAAS login token is a SAML assertion, then the application might process the attribute statements in the assertion.

The WSSMCredential class is defined in com.tivoli.am.fim.wssm.jar, which is included in the `wssm/lib` directory.

Code example

The following code demonstrates accessing this credential. For brevity, import statements and exception handling are not included:

```
Subject subject = WSSubject.getCallerSubject();
Set privateCredentials = subject.getPrivateCredentials(WSSMCredential.class);
Iterator iterator = privateCredentials.iterator();
WSSMCredential wssmCredential = (WSSMCredential) iterator.next();
Element receivedToken = wssmCredential.getReceivedSecurityToken();
Element loginToken = wssmCredential.getLoginSecurityToken();
```

Echo demonstration application

The Echo demonstration application is included with the Web services security management component. The Echo demonstration application is a WebSphere Application Server Web service application that echoes text back to the Echo Web service client to demonstrate the use of the Web services security management component. You can use the Echo demonstration application to verify your Web

services security management component environment configuration, or to demonstrate the functions of the component.

The Echo demonstration application consists of:

Echo Web service definition language (WSDL) file
> A file that describes the operations of the sample Web service application.
>
> File name: `EchoService.wsdl`

The Echo Web service application
> A sample WebSphere Application Server Web service application that implements the Web service defined in `EchoService.wsdl`. The Web service application is packaged as an EAR file, and is installed through the WebSphere Application Server console.
>
> File name: `echoapplication.ear`

The Echo Web service client
> A Java Server Page (JSP) Web application that starts the operations that are defined in `EchoService.wsdl` and are implemented by the Echo Web service application. The client application is packaged as an EAR file and installed through the WebSphere Application Server console.
>
> File name: `echoclientapplication.ear`

The source files for these applications are provided in the `wssm/examples/source/echoapplication` and `wssm/examples/source/echoclientapplication` directories.

Echo application WS-Security configuration

The Echo Web service application is pre-configured with specific WS-Security settings and some settings that are specific to the Web services security management component.

A number of ports are configured, each of which corresponds to the token types that are supported in the demonstration application:

Table 4. Web service token types and associated ports for the Echo application

Token type	Port name
No token	EchoServiceNoToken
Kerberos	EchoServiceKerberos
SAML 1.1	EchoServiceSAML11
SAML 2.0	EchoServiceSAML20
Tivoli Access Manager	EchoServiceTAM
Username	EchoServiceUsername
X.509	EchoServiceX509

In the request consumer section for each of these ports, except EchoServiceNoToken, a required security token, and a caller part are configured.

The token consumer is `com.tivoli.am.fim.wssm.tokenconsumers.WSSMTokenConsumer`

The `sts.call` value for the consumer is set to `true`.

Echo client WS-Security configuration

The Echo Web service client is a Java Server Page (JSP), from which users can interact with the Echo Web service application. The JSP is protected with standard J2EE authentication and authorization. A user must log in with a user name and password that is configured in the WebSphere Application Server user registry, to access the JSP.

A number of ports are configured, each of which corresponds to the token types that are supported in the demonstration application:

Table 5. Web service token types and associated ports for the Echo client

Token type	Port name
No token	EchoServiceNoToken
Kerberos	EchoServiceKerberos
SAML 1.1	EchoServiceSAML11
SAML 2.0	EchoServiceSAML20
Tivoli Access Manager	EchoServiceTAM
Username	EchoServiceUsername
X.509	EchoServiceX509

A security token is configured in the request generator section for each of these ports, except for EchoServiceNoToken. The WSSM token generator is used for all ports, except for EchoServiceNoToken and EchoServiceX509. The standard WebSphere Application Server X.509 token generator is used for EchoServiceX509.

Table 6. Ports and token generators for the Echo client

Port name	Token generator
EchoServiceNoToken	None
EchoKerberos	WSSM token generator
EchoServiceSAML11	WSSM token generator
EchoServiceSAML20	WSSM token generator
EchoSerivceTAM	WSSM token generator
EchoServiceUsername	WSSM token generator
EchoServiceX509	WebSphere Application Server generator: `com.ibm.wsspi.wssecurity.token.X509TokenGenerator`
The WSSM token generator is `com.tivoli.am.fim.wssm.tokenconsumers.WSSMTokenGenerator`.	

The following table describes the callback handlers that are configured for the Echo client.

Table 7. Types and associated callback handlers and properties for the Echo client

Token type	Callback handlers and properties
No token	None

Table 7. Types and associated callback handlers and properties for the Echo client (continued)

Token type	Callback handlers and properties

Kerberos	Callback handler:
	
	    `com.tivoli.am.fim.wssm.callbackhandlers.WSSMTokenGeneratorCallbackHandler`
	
	Properties:
	
	    Name: `xml.callback.handler.class.name`
	
	    Value: `com.tivoli.am.fim.wssm.callbackhandlers.JAASSubjectCallbackHandler`
	
	    Name: `token.callback.handler.class.name`
	
	    Value: `com.tivoli.am.fim.wssm.callbackhandlers.KerberosCallbackHandler`

SAML 1.1	Callback handler:
	
	    `com.tivoli.am.fim.wssm.callbackhandlers.WSSMTokenGeneratorCallbackHandler`
	
	Properties:
	
	    Name: `xml.callback.handler.class.name`
	
	    Value: `com.tivoli.am.fim.wssm.callbackhandlers.JAASSubjectCallbackHandler`
	
	    Name: `token.callback.handler.class.name`
	
	    Value: `com.tivoli.am.fim.wssm.callbackhandlers.SAMLACallbackHandler`

SAML 2.0	Callback handler:
	
	    `com.tivoli.am.fim.wssm.callbackhandlers.WSSMTokenGeneratorCallbackHandler`
	
	Properties:
	
	    Name: `xml.callback.handler.class.name`
	
	    Value: `com.tivoli.am.fim.wssm.callbackhandlers.JAASSubjectCallbackHandler`
	
	    Name: `token.callback.handler.class.name`
	
	    Value: `com.tivoli.am.fim.wssm.callbackhandlers.SAMLACallbackHandler`

Tivoli Access Manager	Callback handler:
	
	    `com.tivoli.am.fim.wssm.callbackhandlers.WSSMTokenGeneratorCallbackHandler`
	
	Properties:
	
	    Name: `xml.callback.handler.class.name`
	
	    Value: `com.tivoli.am.fim.wssm.callbackhandlers.JAASSubjectCallbackHandler`
	
	    Name: `token.callback.handler.class.name`
	
	    Value: `com.tivoli.am.fim.wssm.callbackhandlers.TAMCredentialCallbackHandler`

Username	Callback handler:
	
	    `com.tivoli.am.fim.wssm.callbackhandlers.WSSMTokenGeneratorCallbackHandler`
	
	Properties:
	
	    Name: `xml.callback.handler.class.name`
	
	    Value: `com.tivoli.am.fim.wssm.callbackhandlers.JAASSubjectCallbackHandler`
	
	    Name: `token.callback.handler.class.name`
	
	    Value: `com.tivoli.am.fim.wssm.callbackhandlers.UsernameCallbackHandler`

X.509	Callback handler: `com.ibm.wsspi.wssecurity.auth.callback.X509CallbackHandler`

The `sts.call` value for the generator is set to `false` for EchoServiceSAML11 and EchoServiceSAML20.

Credential handling

To understand the functions that can be demonstrated by the Echo demonstration application, it helps to understand how user credentials are handled by the application as various requests are made.

Note: A user with the name of `wssm-testuser`, and a password of `testonly123` must be configured in both the WebSphere Application Server user registry that is used by the server where the Echo service is running, and in the Tivoli Access Manager user registry.

Login at the Echo client

The user must perform a JAAS login to access the `default.jsp` Web page. The user name and password can be any user that exists in the WebSphere Application Server user registry used by the server where the Echo client is running for this login to succeed. There is no other credential checking performed at the Echo client.

Credentials sent by the Echo client

The following table describes the credentials sent for each token type that can be sent by the Echo client.

Table 8. Credentials sent by the Echo client

Token type	Credentials sent
No token	None.
Kerberos	WSSM generates a SAML 2.0 token internally from the JAAS subject. The STS exchanges it for a Kerberos token.
SAML 1.1	WSSM generates a SAML 1.1 token internally from the JAAS subject. No call is made to the STS.
SAML 2.0	WSSM generates a SAML 2.0 token internally from the JAAS subject. No call is made to the STS.
Tivoli Access Manager	WSSM generates a SAML 2.0 token internally from the JAAS subject. The STS exchanges it for a Tivoli Access Manager token.
Username	WSSM generates a SAML 2.0 token internally from the JAAS subject. The STS exchanges it for a Username token.
X.509	WebSphere Application Server generates an X.509 token. WSSM and the STS are not involved.

Credentials received by the Echo service

The following table describes the credentials received for each token type that can be received by the Echo client.

Table 9. Credentials received by the Echo service

Token type	Credentials received
No token	None.
Kerberos	WSSM receives a Kerberos token and calls the STS to exchange it for a SAML 2.0 token, which is used to perform a login.
SAML 1.1	WSSM receives a SAML 1.1 token and calls the STS to exchange it for Tivoli Access Manager token, which is used to perform a login.
SAML 2.0	WSSM receives a SAML 2.0 token and calls the STS to exchange it for Username token, which is used to perform a login.
Tivoli Access Manager	WSSM receives a Tivoli Access Manager token and calls the STS to exchange it for a SAML 2.0 token, which is used to perform a login.
Username	WSSM receives a Username token and calls the STS to exchange it for a SAML 2.0 token, which is used to perform a login.
X.509	WSSM receives an X.509 token and calls the STS to exchange it for a SAML 1.1 token, which is used to perform a login.

Identity mapping

At the Echo Web service application, there is a many-to-one mapping of the identity so that all calls to the Echo service run under the `wssm-testuser` identity. A Tivoli Access Manager authorization check is used to ensure `wssm-testuser` is authorized to start the Web service. Therefore, the `wssm-testuser` must exist in the Tivoli Access Manager user registry.

Login in at the Echo service

The trust service returns a SAML 2.0 token that contains the `wssm-testuser` identity. The identity is used to perform a JAAS login to the Web service application. Therefore, the `wssm-testuser` must exist in the user registry where the Echo service is running.

Components used in the Echo demonstration application

The Echo demonstration application uses the Web services security management component parts along with some WebSphere Application Server modules to define its WS-Security environment.

The application is designed to demonstrate the operation of the following components:
- WSSM token consumer
- WSSM token generator

The Echo application and client can also be used to demonstrate *no tokens*. The term *no tokens* pertains to an interaction between the client and the service

application without the use of the Web services security management component. In this demonstration, no parts of the Web services security management component are used.

Demonstration of the WSSM token generator

The Echo demonstration application can also be used to demonstrate the use of the WSSM token generator.

The following scenarios can be used to demonstrate the function of the WSSM token generator with the WSSM token consumer:

- Echo client sends a SAML 2.0 token. The WSSM token generator uses the JAASSubjectCallbackHandler to retrieve the user credentials logged in, and package them in a SAML 2.0 assertion token. No call is made to the STS.
- Echo client sends a SAML 1.1 token. The WSSM token generator uses the JAASSubjectCallbackHandler to retrieve the user credentials logged in, and package them in a SAML 1.1 assertion token. No call is made to the STS.
- Echo client sends a Tivoli Access Manager token. The WSSM token generator uses the JAASSubjectCallbackHandler to retrieve the user credentials logged in, and package them in a SAML 2.0 assertion token. Then, it calls the STS to exchange the SAML 2.0 token for a Tivoli Access Manager token.
- Echo client sends a Kerberos token. The WSSM token generator uses the JAASSubjectCallbackHandler to retrieve the user credentials logged in, and package them in a SAML 2.0 assertion token. Then, it calls the STS to exchange the SAML 2.0 token for a Kerberos token.
- Echo client sends a Username token. The WSSM token generator uses the JAASSubjectCallbackHandler to retrieve the user credentials logged in, and package them in a SAML 2.0 assertion token. Then, it calls the STS to exchange the SAML 2.0 token for a Username token.
- Echo client sends an X.509 token. The X509TokenGenerator retrieves an X.509 certificate from a key store and generates an X.509 token.

Demonstration of the WSSM token consumer

The Echo demonstration application can be used to demonstrate the function of the WSSM token consumer.

The following scenarios can be used to demonstrate the use of the WSSM token consumer in combination with the WSSM token generator:

- Echo client sends a SAML 2.0 token. When received by the echo application, the WSSM token consumer accepts the SAML 2.0 token, and calls the STS to exchange it for a Username token. The Username token is then used to log in.
- Echo client sends a SAML 1.1 token. When received by the echo application, the WSSM token consumer accepts the SAML 1.1 token, and calls the STS to exchange it for a Tivoli Access Manager token. The Tivoli Access Manager token is then used to log in.
- Echo client sends a Tivoli Access Manager token. When received by the echo application, the WSSM token consumer accepts the Tivoli Access Manager token, and calls the STS to exchange it for a SAML 2.0 token. The SAML 2.0 token is then used to log in.
- Echo client sends a Kerberos token. When received by the echo application, the WSSM token consumer accepts the Kerberos token, and calls the STS to exchange it for a SAML 2.0 token. The SAML 2.0 token is then used to log in.

- Echo client sends a Username token. When received by the echo application, the WSSM token consumer accepts the Username token and calls the STS to exchange it for a SAML 2.0 token. The SAML 2.0 token is then used to log in.
- Echo client sends an X.509 token. When received by the echo application, the WSSM token consumer accepts the X.509 token and calls the STS to exchange it for a SAML 1.1 token. The SAML 1.1 token is then used to log in.

See "Demonstration of the WSSM token generator" on page 24 for more information.

Chapter 2. Configuring WebSphere Application Server

To extend the capabilities of WebSphere Application Server, you can enable Tivoli Federated Identity Manager Web services security management.

About this task

Before performing these tasks, be sure that you have completed all installation and configuration tasks described in the *IBM Tivoli Federated Identity Manager Installation Guide*. As described in those guides, the prerequisite configuration tasks for WebSphere Application Server include:

1. Installing WebSphere Application Server, including creating an application server profile
2. Enabling global security, including configuring a user registry

See the IBM WebSphere Application Server Information Center for assistance with these tasks:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp

Note: Running Web services security management on the embedded version of WebSphere Application Server is not supported.

Note: WebSphere Application Server 7.0.0.7 is required for JAX-WS support with Tivoli Federated Identity Manager.

Starting the WebSphere Application Server administrative console

Open your Web browser to the WebSphere Application Server administrative console.

Procedure

1. Open your Web browser.
2. Go to the WebSphere Application Server administrative console associated with the application server where the Web services security management component is installed. For example, if the administrative console is running on the same system as the Web browser and was installed using port number 9060, then use the following URL:

   `http://localhost:9060/ibm/console`
3. When prompted, enter the password that corresponds to the administrative user ID.
4. Click **Log in**.

Setting WebSphere variables

Several environment variables must be defined in WebSphere Application Server for the application to use Web services security management.

Procedure

1. Start the WebSphere Application Server administrative console and log on, if necessary.

Note: Ensure that you are using the administrative console associated with the application server where the Web services security management component, and the application server profiles for the applications you are securing with the Web services security management component, are installed.

2. Click **Environment** > **WebSphere Variables** to open the WebSphere Variables page.

3. Select a scope and click **New** to add an environment variable that specifies the installation directory for the Web services security management component of Tivoli Federated Identity Manager. For a WebSphere Cluster, use the **Node** scope and define the variable for each Node in the Cluster.

4. Enter the following information to define a variable called ITFIM_WSSM.

   a. In the **Name** field, enter ITFIM_WSSM.

   b. In the **Value** field, enter the name of the directory where the Web services security management component is installed. The default values are:

   UNIX and Linux
   /opt/IBM/FIM/wssm

   Windows
   C:\Program Files\IBM\FIM\wssm

   c. In the **Description** field, enter a description of the contents or purpose of the variable for future reference. For example, `Installation directory for the Web services security component of Tivoli Federated Identity Manager`.

   d. Click **OK** to add the variable.

5. In the Messages pane at the top of the WebSphere Variables window, click **Save** to commit your changes.

Configuring WebSphere shared library

Define a shared library in WebSphere Application Server so that the application can use Web services security management.

Procedure

1. Start the WebSphere Application Server administrative console and log on, if necessary.

   Note: Ensure that you are using the administrative console associated with the application server where the Web services security management component is installed.

2. Click **Environment** > **Shared Libraries** to open the Shared Libraries page.

3. Select a scope and click **New** to specify the shared library needed by Tivoli Federated Identity Manager. For a WebSphere Cluster, use the **Node** scope and define the variable for each Node in the Cluster.

4. Enter the following information to define the necessary shared library.

   a. In the **Name** field, enter ITFIM_WSSM. This name is needed later when configuring the class loader.

   b. In the **Description** field, enter a description of the contents or purpose of the shared library for future reference. For example, `Shared libraries needed by the Web services security management component of Tivoli Federated Identity Manager`.

   c. In the **Classpath** field, enter the following information:

```
${ITFIM_WSSM}/etc
${ITFIM_WSSM}/lib/com.tivoli.am.fim.wssm.jar
${ITFIM_WSSM}/lib/com.tivoli.am.fim.common.jar
${ITFIM_WSSM}/lib/com.tivoli.am.fim.management.jar
${ITFIM_WSSM}/lib/com.tivoli.am.fim.war.sts.stubs.client.jar
${ITFIM_WSSM}/lib/com.tivoli.am.fim.midmgr.jar
${ITFIM_WSSM}/lib/com.tivoli.am.fim.sts.jar
${ITFIM_WSSM}/lib/itfim-locale-msgs.jar
```

Note: This list also can be copied from the following file:

UNIX and Linux
> `/opt/IBM/FIM/wssm/etc/wssm.classpath`

Windows
> `C:\Program Files\IBM\FIM\wssm\etc\wssm.classpath`

d. Leave the **Native Library Path** field blank.

e. Click **OK** to add the shared library.

5. In the Messages pane at the top of the Shared Libraries window, click **Save** to commit your changes.

Configuring for a cluster environment

Copy the Web services security management directory and files to each node in the cluster, to use Web services security management in a WebSphere Application Server cluster environment.

Procedure

1. Create a *WSSM_HOME* directory on each node in the cluster. By default, this directory is:

UNIX and Linux
> /opt/IBM/FIM/wssm

Windows
> C:\Program Files\IBM\FIM\wssm

2. Copy the contents of the *WSSM_HOME* directory of the Deployment Manager to each node in the cluster.

Associating the shared library with the application

You must associate the shared library with the web service provider and requester application before you can use it.

Procedure

1. If you use WebSphere Application Server, version 6.1, start the administrative console and log in, if necessary.

   Note: Ensure that you are using the administrative console associated with the application server where the Web services security management component is installed.

2. Click **Applications** > **Enterprise Applications** > **application_name** > **Shared library references** in the console navigation tree to access the Shared library references page.

3. In the Shared library references page, select an application or the module that you want to associate with the shared library.

4. Click **Reference shared libraries**.

5. In the Shared Library Mapping page, select the ITFIM_WSSM shared library in the **Available** list. Click **>>** to add them to the Selected list. Then, click **OK**.

6. In the Shared library references page, click **OK**.

7. Save the changes to the configuration.

   Note: If you are using other WebSphere Application Server versions, see the following topics:
   - WebSphere Application Server, version 8.0 - Associating shared libraries with applications or modules
   - WebSphere Application Server, version 7.0 - Associating shared libraries with applications or modules
   - WebSphere Application Server, version 6.0.x - Associating shared libraries with applications

Configuring a SAML login module for WebSphere Application Server

The standard system login modules provided by WebSphere Application Server do not support using a SAML assertion to log on. Perform this task to use the SAML login module provided with the Web services security management component.

Procedure

1. Start the WebSphere Application Server administrative console and log on, if necessary.

   Note: Ensure that you are using the administrative console associated with the application server where the Web services security management component is installed.

2. Click **Security** > **Secure administration, applications, and infrastructure**.

3. In the Authentication section, expand the **Java Authentication and Authorization Service** container, and click **System logins**.

4. Click **New** to define a new login configuration.

5. In the **Alias** field, enter `itfim.wssm.samla`, and then click **Apply**.

   Note: All system login configurations have an implied prefix of `system`. This system login configuration might be referred to as `system.itfim.wssm.samla`.

6. In the Additional Properties section, click **JAAS login modules**.

7. Click **New** to define a new system login module.

8. In the **Module class name** field, enter: `com.tivoli.am.fim.wssm.loginmodules.SAMLALoginModule`, and then click **OK**.

9. In the Messages pane at the top of the window, click **Save** to commit your changes.

10. Restart the application server.

Configuring a Tivoli Access Manager login module for WebSphere Application Server

The standard system login modules provided by WebSphere Application Server do not support the use of a Tivoli Access Manager assertion to log on. Perform this task to use the Tivoli Access Manager login module provided with the Web services security management component.

Before you begin

The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

Procedure

1. Start the WebSphere Application Server administrative console and log on, if necessary.

   Note: Ensure that you are using the administrative console associated with the application server where the Web services security management component is installed.

2. Click **Security** > **Secure administration, applications, and infrastructure**.

3. In the Authentication section, expand the **Java Authentication and Authorization Service** container, and click **System logins**.

4. Click **New** to define a new login configuration.

5. In the **Alias** field, enter `itfim.wssm.tam`, and then click **Apply**.

   Note: All system login configurations have an implied prefix of `system`. This system login configuration might be referred to as `system.itfim.wssm.tam`.

6. In the Additional Properties section, click **JAAS login modules**.

7. Click **New** to define a new system login module.

8. In the **Module class name** field, enter: `com.tivoli.am.fim.wssm.loginmodules.TAMLoginModule`, and then click **OK**.

9. In the Messages pane at the top of the window, click **Save** to commit your changes.

10. Restart the application server.

Configuring the Tivoli Access Manager Login Module with echoapplication

Use the Tivoli Access Manager Login Module with the echoapplication program, the test application shipped with Tivoli Federated Identity Manager.

Before you begin

The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

Procedure

1. Start the WebSphere Application Server administrative console and log on, if necessary.

   Note: Ensure that you are using the administrative console associated with the application server where the Web services security management component is installed.

2. Click **Security** > **Secure administration, applications, and infrastructure**.

3. In the Authentication section, expand the **Java Authentication and Authorization Service** container, and click **System logins**.

4. Add a new entry for `itfim.wssm.tam`, then select the new module and specify the JAAS login modules with Module class name:`com.tivoli.am.fim.wssm.loginmodules.TAMLoginModule`

5. Import the **echoapplication.ear** into Rational Application Developer or WebSphere Application Server Toolkit. Then modify:

   a. **Extension**

   For EchoServiceUsername port component binding, edit the caller part and set:

   ```
   URI: <nothing>
   Local name: http://ibm.com/2004/01/itfim/ivcred
   ```

   b. **Binding**

   In the EchoServiceUsername port component binding, change the Token consumer `jaas.config` name to:

   ```
   system.itfim.wssm.tam
   ```

   Save the new application for later installation into WebSphere Application Server.

6. In `<wssm_install_root>/wssm.properties`, add:

   ```
   pdjrte.configuration=path-to-TAM-config-file>
   ```

   For example, the file can be the same file used for the Tivoli Federated Identity Manager runtime configuration:

   ```
   /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/config/itfim/
   dp-autotest-server1/nodes/dp-autotestNode01Cell/
   dp-autotestNode01/server1/amconfig.conf
   ```

   This setting is used by the Tivoli Access Manager login module.

7. In the Tivoli Federated Identity Manager trust service, use the Tivoli Access Manager credential module to set the application trust chain to issue Tivoli Access Manager credentials.

   If an application trust chain was previously created, then change the application trust chain to issue Tivoli Access Manager credentials rather than SAML 2.0 credentials. For example, edit the application chain and change the last module from SAML 2.0 to Tivoli Access Manager credential module.

8. Install the new version of the application into the WebSphere Application Server.

9. Restart WebSphere Application Server.

10. Using the echoclientapplication, select **UsernameToken**.

11. Click **WhoAmI**. The JAAS subject is shown with a private WSSMCredential containing the BinarySecurityToken representing the Tivoli Access Manager credential.

Configuring a Username login module for WebSphere Application Server

The Web services security management component must use its own Username JAAS configuration instead of the WebSphere Application Server Username JAAS configuration. This is because it uses different callbacks between the Web services security management token consumer and the Username login module. Perform this task to use the Username login module provided with the Web services security management component.

Procedure

1. Start the WebSphere Application Server administrative console and log on, if necessary.

   Note: Ensure that you are using the administrative console associated with the application server where the Web services security management component is installed.

2. Click **Security** > **Secure administration, applications, and infrastructure**.

3. In the Authentication section, expand the **Java Authentication and Authorization Service** container, and click **System logins**.

4. Click **New** to define a new login configuration.

5. In the **Alias** field, enter `itfim.wssm.username`, and then click **Apply**.

   Note: All system login configurations have an implied prefix of `system`. This system login configuration might be referred to as `system.itfim.wssm.username`.

6. In the Additional Properties section, click **JAAS login modules**.

7. Click **New** to define a new system login module.

8. In the **Module class name** field, enter: `com.tivoli.am.fim.wssm.loginmodules.UsernameLoginModule`, and then click **OK**.

9. In the Messages pane at the top of the window, click **Save** to commit your changes.

10. Restart the application server.

Configuring Java 2 security

If you have selected the **Enforce Java 2 security** box in the WebSphere Application Server Global Security settings, you must configure the `library.policy` file for the Tivoli Federated Identity Manager node.

Before you begin

To configure the library.policy file, you must add to it the permissions that are specified in the sample `wssm.policy` file that was installed with the Web services security management component files.

About this task

To perform this task, you must be familiar with editing WebSphere Application Server policy files using the WebSphere Application Server Policy Tool. See the following topics in the WebSphere Application Server Information Center at http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp:

- Configuring Java 2 policy files
- library.policy file permissions
- Using the Policy Tool to edit policy files

After reviewing these topics, continue with the following procedure:

Procedure

1. Open a command prompt on the system where Tivoli Federated Identity Manager is installed.
2. Navigate to the directory that contains the files for the Web services security management component. The files can be found in the following default locations:

   UNIX and Linux
   /opt/IBM/FIM/wssm/etc

   Windows
   C:\Program Files\IBM\FIM\wssm\etc

3. Locate the wssm.policy file and open it in a text editor.
4. Make any changes to the paths as required for your environment. Then save the file. Keep the file open so that you can use its contents to modify the library.policy file on the WebSphere Application Server.
5. Log in to the WebSphere Application Server and open a command prompt.
6. Using the Policy Tool as described in the WebSphere Application Server Information Center, modify the library.policy file with the information from the wssm.policy file. The library.policy file can be found in the following default locations:

   AIX® - /usr/IBM/WebSphere/AppServer/profiles/AppSrv01/config/cells/
   fim6ode01Cell/nodes/fim6Node01

   UNIX and Linux
   /opt/IBM/WebSphere/AppServer for UNIX/profiles/AppSrv01/config/
   cells/fim6ode01Cell/nodes/fim6Node01

   Windows
   C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\
   config\cells\fim6ode01Cell\nodes\fim6Node01

   Note: These path names have been formatted to fit this page.
7. Restart the WebSphere Application Server.

Securing the connection to the trust service

You can secure the connection between the Web services security management component and the trust service using HTTPS.

Before you begin

Before beginning this task, you need to know which SSL repository is being used to secure the HTTPS port.

Procedure

1. Start the WebSphere Application Server administrative console and log in, if necessary.

2. Click **Servers** > **Application servers** > **server1** > **Web Container Settings** > **Web container transport chains**.

   Note: The **server1** selection shown in this step might have a different name in your environment.

3. Select the chain for the port being used and then specify the SSL repertoire being used for the SSL channel. For example: NodeDefaultSSLSettings

4. Update the wssm.properties file to identify the repertoire. Refer to "Modifying the wssm.properties file" on page 39.

Controlling trust service authorization access

Learn about the different scenarios for controlling access to the trust service.

You can choose from the following three scenarios for controlling access to the trust service:

- Bypass WebSphere authentication for access to the trust service, relying on message level authentication for Web services messages.
- Use WebSphere authentication for Web services messages.
- Use a mix of both message level authentication and WebSphere authentication.

Using message level authentication only

Use the WebSphere administration console to map the TrustClientRole.

Before you begin

1. Log on to the WebSphere administration console and go to the properties for the ITFIMRuntime application.

2. Click **Map Security Roles to users/groups**.

3. Map the **TrustClientRole** to **Everyone**.

You are not required to authenticate to WebSphere Application Server before sending messages to the trust service. You must verify that your trust service modules provide adequate authentication and authorization of Web services messages.

Using WebSphere authentication mechanisms with message-level authentication

Learn how to use WebSphere authentication mechanisms with Web services message-level authentication.

Before you begin

If you must use a mixture of WebSphere authentication mechanisms with Web services message-level authentication, use two separate URLs and two separate J2EE roles to control access to the trust service. Clients that can provide authentication information directly to WebSphere use one URL, while clients that provide only message-level authentication use the other URL.

IBM Tivoli Federated Identity Manager configures two URLs and two J2EE roles for access to the security token service. Access to the /TrustServer/SecurityTokenServiceProtected URL is granted to the TrustClientRole. Access to the/TrustServer/SecurityTokenService URL is granted to the client

TrustClientInternalRole. Clients can choose their own authentication mechanism given the access.

Using WebSphere authentication to access the trust service

Learn to use WebSphere authentication to give access to the trust service

Before you begin

Follow these steps to give appropriate access to the trust service.

1. Log on to the WebSphere administration console, and go to the properties for the ITFIMRuntime application.
2. Click **Map Security Roles to users/groups**.
3. Map TrustClientRole to those users or groups you want to give access to the trust service.

Restarting WebSphere Application Server

WebSphere Application Server can be restarted in a number of ways.

Before you begin

After performing some operations in either the WebSphere Application Server administrative console or the Tivoli Federated Identity Manager administration console, you must restart WebSphere Application Server to apply your changes.

Using the command line

Script and batch files are provided for stopping and starting WebSphere Application Server.

About this task

Restart the WebSphere Application Server using the command line:

Procedure

1. Open a command prompt.
2. Locate the server scripts in the WebSphere Application Server installation directory and run them, specifying the name of the application server to stop and start. The following commands assume that WebSphere Application Server was installed in the default location:

   AIX

```
/usr/IBM/WebSphere/AppServer/bin/stopServer.sh server_name -profileName profile_name  \
-username user_name -password password
/usr/IBM/WebSphere/AppServer/bin/startServer.sh server_name -profileName profile_name  \
-username user_name -password password
```

   Linux and Solaris

```
/opt/IBM/WebSphere/AppServer/bin/stopServer.sh server_name \
 -profileName profile_name  \
-username user_name -password password \
/opt/IBM/WebSphere/AppServer/bin/startServer.sh server_name -profileName profile_name  \
-username user_name -password password
```

   Windows

```
"C:\Program Files\IBM\WebSphere\AppServer\bin\stopServer.bat" server_name -profileName profile_name
-username user_name -password password
```

```
"C:\Program Files\IBM\WebSphere\AppServer\bin\startServer.bat" server_name -profileName profile_name
-username user_name -password password
```

For example, to stop and start an application called BankApp on a Windows system, you might use the following two commands:

```
"C:\Program Files\IBM\WebSphere\AppServer\bin\stopServer.bat" server1 -profileName BankApp
  -username wasadmin -password was60pwd
```

```
"C:\Program Files\IBM\WebSphere\AppServer\bin\startServer.bat" server1 -profileName BankApp
-username wasadmin -password was60pwd
```

Using the Tivoli Federated Identity Manager administration console

Procedure

1. After performing a task in the Tivoli Federated Identity Manager administration console, message FBTCON197W might be shown indicating that WebSphere Application Server must be restarted.
2. Click **Restart WebSphere** to stop and restart the current application server.

   Note: After starting the application server, wait until WebSphere Application Server has restarted before performing additional actions.

Checking WebSphere Application Server status

After starting WebSphere Application Server, check that the server is running before performing additional tasks.

Using the command line

Script and batch files are provided for checking the status of WebSphere Application Server.

About this task

To check the status of WebSphere Application Server using the command line:

Procedure

1. Open a command prompt.
2. Locate the serverStatus script or batch file in the WebSphere Application Server installation directory. Run it, specifying the profile name and application server name. The following commands assume that WebSphere Application Server was installed in the default location:

   AIX

```
/usr/IBM/WebSphere/AppServer/bin/serverStatus.sh server_name  -profileName profile_name
```

   Linux and Solaris

```
/opt/IBM/WebSphere/AppServer/bin/stopServer.sh server_name  -profileName profile_name
```

   Windows

```
C:\Program Files\IBM\WebSphere\AppServer\bin\serverStatus.bat server_name  -profileName profile_name
```

Chapter 3. Updating configuration batch and script files

A number of script and batch files must be updated to reflect your environment before you can enable application security.

Before you begin

Configuration batch and script files can be run against WebSphere Application Server V6.0.2, V6.1, V7.0, and V8.0.

Specify the WebSphere Application Server class path before you run your script using commenting. WebSphere Application Server V6.0.2, V6.1, V7.0, and V8.0 have different class paths.

In the following example, the wsdl2tam.sh script is set to run against WebSphere Application Server V6.1:

```
# The required WAS 6.0.2 jars.

# export WAS_JARS=$WAS_HOME/lib/j2ee.jar:$WAS_HOME/lib/wsdl4j.jar

# The required WAS 6.1 jars.

export WAS_JARS=$WAS_HOME/lib/j2ee.jar:$WAS_HOME/plugins/
 com.ibm.ws.runtime_6.1.0.jar

# The required WAS 7.0 jars.

# export WAS_JARS=$WAS_HOME/lib/j2ee.jar:$WAS_HOME/plugins/com.ibm.ws.prereq.wsdl4j.jar
```

To run this script against WebSphere Application Server V6.0.2, uncomment the earlier WAS-JARS export line and comment the current WAS_JAR export line; for example:

```
# The required WAS 6.0.2 jars.

  export WAS_JARS=$WAS_HOME/lib/j2ee.jar:$WAS_HOME/lib/wsdl4j.jar

# The required WAS 6.1 jars.

# export WAS_JARS=$WAS_HOME/lib/j2ee.jar:$WAS_HOME/plugins/
 com.ibm.ws.runtime_6.1.0.jar

# The required WAS 7.0 jars.

# export WAS_JARS=$WAS_HOME/lib/j2ee.jar:$WAS_HOME/plugins/com.ibm.ws.prereq.wsdl4j.jar
```

Modifying the wssm.properties file

Update the wssm.properties file to reflect the appropriate values for WebSphere Application Server and Tivoli Federated Identity Manager. You must update this file before using the trust service.

Procedure

1. Go to the *WSSM_HOME* directory. By default, this directory is:

   UNIX and Linux
   /opt/IBM/FIM/wssm/etc

Windows

           `C:\Program Files\IBM\FIM\wssm\etc`

2. Make a backup copy of the `wssm.properties` file.

3. Verify that the values specified are correct for your environment.

    a. Set the `sts.url` property. For example: `sts.url=http://localhost:9080/` `TrustServer/SecurityTokenService`. If the connection between the trust service and the Web services security management component is secured with HTTPS, an example would be `sts.url=https://localhost:9443/` `TrustServer/SecurityTokenService`

    b. Optional: If you are using HTTPS to secure the connection between the Web services security management component and the trust service, set the `ssl.configuration` property. For example: `ssl.configuration=fim6Node01/` `DefaultSSLSettings`

       Note: Use WebSphere Application Server administration console to make sure that the SSL repository has also been identified.

    c. Optionally, if you are using HTTP basic authentication to secure the connection between the Web services security management component and the trust service, set the authorization.username and authorization.password properties. For example: `authorization.username=wssm` and `authorization.password=testonly`.

    d. Optionally, if your application is creating or using Tivoli Access Manager security tokens, set the pdjrte.configuration property to the Tivoli Access Manager Java Runtime (PDJrte) configuration file. For example: `pdjrte.configuration=/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/` `config/itfim/mg8lnx10-FIMCluster/nodes/mg8lnx10Cell01/mg8lnx11Node01/` `mg8lnx11/amconfig.conf`

4. Save the changes to the file. You do not need to restart WebSphere Application Server after these changes.

Sample `wssm.properties` file

The Web services security management component uses a `wssm.properties` file that must be modified.

```
# The following are the configuration properties for Web Services Security Management (WSSM).
#

# The Trust Service URL
sts.url=http://localhost:9080/TrustServer/SecurityTokenService

# The WAS SSL configuration repertoire name (only required for HTTPS connections to the Trust Service)
ssl.configuration=

# The optional username, passed in the HTTP authorization header, identifying WSSM to the Trust Service
authorization.username=

# The optional password, passed in the HTTP authorization header, associated with the username
authorization.password=

# The location of a TAM configuration file (created with SrvSslCfg). This is used by the TAMLoginModule.
#pdjrte.configuration=
```

Figure 2. Sample `wssm.properties` file

Modifying the `wsdl2tam` file

The `wsdl2tam` script or batch file must be updated to reflect the appropriate values for Java, WebSphere Application Server, Tivoli Federated Identity Manager, and the Web services security management component.

Procedure

1. Go to the *WSSM_HOME* directory. By default, this directory is:

   UNIX and Linux
   > `/opt/IBM/FIM/wssm/etc`

   Windows
   > `C:\Program Files\IBM\FIM\wssm\etc`

2. Make a backup copy of the `wsdl2tam.sh` or `wsdl2tam.bat` file. In general, the `wsdl2tam.sh` file is used on UNIX, Linux systems, and Windows systems using a UNIX shell, such as **bash**. The `wsdl2tam.bat` file is used on Windows systems.

3. Open the file in a text editor and verify that the values specified are correct for your environment.

   a. *JAVA_HOME* is the directory for the Java runtime environment being used by your WebSphere application.

   b. *WAS_HOME* is the directory where WebSphere Application Server is installed.

   c. *FIM_HOME* is the directory where Tivoli Federated Identity Manager is installed.

   d. *WSSM_HOME* is the directory where the Web services security management component of Tivoli Federated Identity Manager is installed.

4. Save the changes to the file.

Sample `wsdl2tam.sh` file

The Web services security management component uses the `wsdl2tam.sh` file, which must be modified. The `wsdl2tam.sh` file is used on UNIX, and Linux systems. The file also can be used on Windows systems running a UNIX shell program, such as **bash**.

The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

```
#!/bin/sh

#
# Runs the WSDL2TAM tool for taking WSDL documents and generating script for creating TAM protected objects.
#
# NB. Requires Java 1.4.2 or better. Ensure WAS_JARS is set for the correct version of WAS.
#

# Make sure the Java home directory is correct.

export JAVA_HOME=/opt/IBM/WebSphere/AppServer/java/jre

# Make sure the WAS home directory location is correct.

export WAS_HOME=/opt/IBM/WebSphere/AppServer

# Make sure the FIM home directory location is correct.

export FIM_HOME=/opt/IBM/FIM

# The WSSM home directory.

export WSSM_HOME=$FIM_HOME/wssm

# The required WAS 6.0.2 jars.

# export WAS_JARS=$WAS_HOME/lib/j2ee.jar:$WAS_HOME/lib/wsdl4j.jar

# The required WAS 6.1 jars.

export WAS_JARS=$WAS_HOME/lib/j2ee.jar:$WAS_HOME/plugins/com.ibm.ws.runtime_6.1.0.jar

# The required WSSM jars.

export WSSM_JARS=$WSSM_HOME/lib/itfim-wssm-common.jar:$WSSM_HOME/lib/itfim-wssm-mgmt.jar

export CLASSPATH=$WSSM_HOME:$WSSM_JARS:$WAS_JARS:$CLASSPATH

$JAVA_HOME/bin/java -Djava.ext.dirs="$JAVA_HOME/lib/ext:$WAS_HOME/lib"
 -Djava.util.logging.config.file=$WSSM_HOME/
 logging.properties com.tivoli.am.fim.wssm.management.WSDL2TAM $*
```

Note: Some lines of text in the figure have been split for readability.

Figure 3. Sample `wsdl2tam.sh` file

Sample `wsdl2tam.bat` file

The Web services security management component uses the `wsdl2tam.bat` file, which must be modified. The `wsdl2tam.bat` file is used on Windows systems.

The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

```
@echo off

rem Runs the WSDL2TAM tool for taking WSDL documents and generating script for creating TAM protected objects.
rem
rem NB. Requires Java 1.4.2 or better. Ensure WAS_JARS is set for the correct version of WAS.

setlocal

rem Make sure the Java home directory is correct.

set JAVA_HOME=C:\Program Files\IBM\WebSphere\AppServer\java\jre

rem Make sure the WAS home directory location is correct.

set WAS_HOME=C:\Program Files\IBM\WebSphere\AppServer

rem Make sure the FIM home directory location is correct.

set FIM_HOME=C:\Program Files\IBM\FIM

rem The WSSM home directory.

set WSSM_HOME=%FIM_HOME%\wssm

rem The required WAS 6.0.2 jars.

rem set WAS_JARS=%WAS_HOME%/lib/j2ee.jar;%WAS_HOME%/lib/wsdl4j.jar

rem The required WAS 6.1 jars.

set WAS_JARS=%WAS_HOME%/lib/j2ee.jar;%WAS_HOME%/plugins/com.ibm.ws.runtime_6.1.0.jar

rem The required WSSM jars.

set WSSM_JARS=%WSSM_HOME%/lib/itfim-wssm-common.jar;%WSSM_HOME%/lib/itfim-wssm-mgmt.jar

set CLASSPATH=%WSSM_JARS%;%WAS_JARS%;%CLASSPATH%

"%JAVA_HOME%\bin\java" -Djava.ext.dirs="%JAVA_HOME%/lib/ext;%WAS_HOME%/lib"
 -Djava.util.logging.config.file="%WSSM_HOME%"/logging.properties
 com.tivoli.am.fim.wssm.management.WSDL2T
 AM %*

endlocal
```

Note: Some lines of text in the figure have been split for readability.

Figure 4. Sample `wsdl2tam.bat` *file*

Enabling logging for wsdl2tam

You can monitor the activities of the **wsdl2tam** tool by enabling logging for it.

Procedure

1. Go to the *WSSM_HOME* directory. By default, this directory is:

   UNIX and Linux
   `/opt/IBM/FIM/wssm/etc`

   Windows
   `C:\Program Files\IBM\FIM\wssm\etc`

2. Make a backup copy of the `logging.properties` file.
3. To enable logging for the **wsdl2tam** tool, set the `.level` property to ALL.

4. Save the changes to the file.

Chapter 4. Configuring Tivoli Access Manager

Several configuration tasks are required if you plan to use Tivoli Access Manager for e-business as an authorization solution with the Web services security management component.

Before you begin

The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

Object space

Tivoli Access Manager can provide an authorization solution for your Web service application. To use this function, you must create a protected object space and objects for your Web service application in Tivoli Access Manager.

The Web services security management component provides the `wsdl2tam` command that can automate this object space creation. The command reads the WSDL file associated with a Web service application and creates a script file that you use with the `pdadmin` command.

This process creates objects that represent the corresponding Web services in your application. The object names are based on the names used in the WSDL file. Access control lists (ACLs) can be applied at different levels in the object hierarchy to provide coarse-grained or fine-grained access to the Web service application.
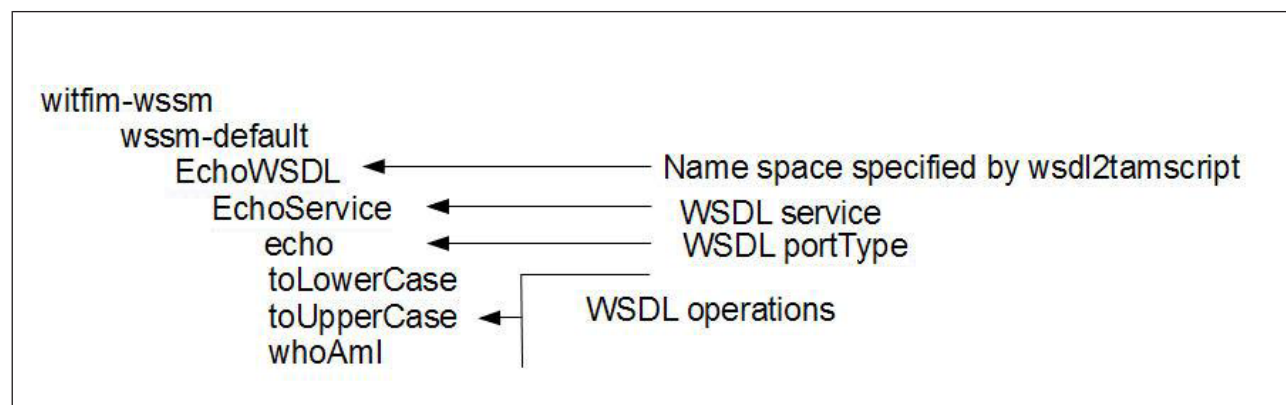
Figure 5. The object space created in Tivoli Access Manager for Web services security management component managed Web services

Figure 5 shows the object space that is created for the Echo demonstration application provided with Tivoli Federated Identity Manager. If an ACL granting user access is added to the `EchoService` object, that user can access any of the operations provided by the Echo demonstration application.[1]

---

1. The full name of the `EchoService` object used in this example is:

   `/itfim-wssm/wssm-default/EchoWSDL/EchoService/EchoService`

In addition to ACLs, other Tivoli Access Manager controls, such as protected object policies (POPs) and authorization rules, also can be applied to these objects to control access to the Web services based on the day of the week, the time of day, and so on.

Creating protected objects

If you plan to use the AuthorizationSTSModule with Tivoli Access Manager, you must create a name space and protected objects in Tivoli Access Manager that correspond to your WSDL file. By running the **wsdl2tam** command, you can create a script file that you can use with **pdadmin** to automate the creation of the space and objects.

Before you begin

The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

The **wsdl2tam** script or batch file must be modified before performing this task. Both the Tivoli Access Manager policy server and authorization server must be running and the **pdadmin** command must be accessible in the command path.

Procedure

1. Open a command prompt on the system where WebSphere Application Server is installed.
2. Go to the directory that contains the files for the Web services security management component. The files can be found in the following default location:

   UNIX and Linux
   > /opt/IBM/FIM/wssm/etc

   Windows
   > C:\Program Files\IBM\FIM\wssm\etc

3. Run the modified **wsdl2tam** script or batch file specifying the following identification:
   - A unique logical name for the WSDL document (such as BankingWSDL)
   - The name of the script file to be created (such as `tamscript.txt`)
   - The URL or file name with the WSDL file for the application (such as `ExampleBanking.wsdl`)

   UNIX and Linux
   > `./wsdl2tam.sh -n BankingWSDL -tam tamscript.txt ExampleBanking.wsdl`

   > **Note:** When running this script on a Solaris system, you must use the **ksh** shell instead of the default shell (**/bin/sh**).

   Windows
   > `wsdl2tam.bat -n BankingWSDL -tam tamscript.txt ExampleBanking.wsdl`

4. Open the script file and review its content to make sure that it is correct.

5. Save and close the file.
6. Open a command prompt on the system where Tivoli Access Manager is installed.
7. Next, use the **pdadmin** command to run the script file, such as `tamscript.txt`, and create the protected object space. For example, use the following command:

```
pdadmin -a sec_master -p testonly tamscript.txt
```

   Note: The script attempts to create all the objects in the hierarchy. If some of these objects exist, messages regarding the existing objects show. You can ignore these messages.
8. Optional: Verify that the protected object space was created by using the **pdadmin** command or Web Portal Manager.

wsdl2tam reference

The wsdl2tam command reference creates a protected object space in Tivoli Access Manager for objects defined in a WSDL file to be used with the AuthorizationSTSModule and Tivoli Access Manager.

The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

Syntax

```
wsdl2tam{.sh|.bat} { -n unique_name -tam name_of_output_file WSDL_URL_or_file }
```

Options

-n *unique_name*
> A unique logical name for the WSDL document. This name is used as part of the name for the object space that is created.

-tam *name_of_output_file*
> The name of the script file to be created.

WSDL_URL_or_file_name
> A URL or file name where the WSDL definition for the Web service is located.

Authorization

You must have a valid administrator user ID and password on the server and must have the required authority to perform the task.

Example: Creating protected objects in Tivoli Access Manager

Define protected objects called `CustomerBanking` based on the WSDL file located in the current directory on UNIX or Linux:

```
./wsdl2tam.sh -n CustomerBanking -tam tamscript.txt Banking.wsdl
```

Defining the access control policy

An access control policy must be defined for the Tivoli Federated Identity Manager access control objects.

Before you begin

The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

Procedure

1. Open a command prompt on the system where Tivoli Access Manager is installed.
2. Use the **pdadmin** command to define your preferred access control policy in Tivoli Access Manager.

Example

For example, if you want to create an access control group called BankUsers that has permission to start any Web service in an application, and define a user called wssm-bankuser that is a member of that group, use the following **pdadmin** commands:

```
pdadmin -a sec_master -p password
user create wssm-bankuser cn=wssmbankuser,c=us wssm bankuser initial_user_password
user modify wssm-bankuser account-valid yes
group create BankUsers cn=BankUsers,c=us BankUsers
group modify BankUsers add wssm-bankuser
acl create BankUsers
acl modify BankUsers set group BankUsers T[WebService]i
acl attach /itfim-wssm/wssm-default/BankingWSDL BankUsers
```

The two user commands create the user and activate the user account. The next two group commands create an access control group and assign the user to the group. Finally, the two acl commands create an access control list and attach that list to all of the access control objects associated with the Echo demonstration application. The result is that the wssm-bankuser user ID has permission to start any BankingWSDL Web service.

What to do next

See the *IBM Tivoli Access Manager for e-business Administration Guide* for additional details on ACLs.

Chapter 5. Configuring keystores and certificates

You might need to configure keys or certificates before configuring the token modules in your module chains, depending on your application requirements, and the types of tokens you use with the Web services security management component.

Before you begin

The Tivoli Federated Identity Manager service for managing keys is called the *key encryption and signing service*, or simply, *key service*. The key service is responsible for managing certificates and keys, and for performing cryptographic operation. If you use keys or certificates with your tokens, you should import your keys or certificates into this key service.

See the *IBM Tivoli Federated Identity Manager Administration Guide* for instructions on managing keys and certificates. Use the Tivoli Federated Identity Manager administration console to configure the key service and the token modules.

You can use the Tivoli Federated Identity Manager management console to generate self-signed certificates, generate a certificate request to be signed by a Certificate Authority and receive the signed certificate, and store certificates from an SSL connection.

About this task

The token modules that can be used with the Web services security management component have the following key or certificate configuration options:

SAML10STSModule and SAML11STSModule
> **Validate mode:**
> - Validation key that the partner must use. Specify the keystore, keystore password, and a key or certificate to use.
>
> **Exchange mode:**
> - Signing key used to sign outgoing tokens. Specify the keystore, keystore password, and a key or certificate to use.

SAML20STSModule
> **Validate mode:**
> - Validation key that the partner must use. Specify the keystore, keystore password, and a key or certificate to use.
> - Decryption key used to decrypt encrypted messages. Specify the keystore, keystore password, and a key or certificate to use.
>
> **Exchange mode:**
> - Signing key used to sign outgoing tokens. Specify the keystore, keystore password, and a key or certificate to use.
> - Encryption key used to encrypt assertions. Specify the keystore, keystore password, and a key or certificate to use.

UsernameTokenSTSModule
> **Validate mode:**

- Validation key that the partner must use. Specify the keystore, keystore password, and a key or certificate to use.

Exchange mode:
- Signing key used to sign outgoing tokens. Specify the keystore, keystore password, and a key or certificate to use.

KerberosSTSModule
No key or certificate configuration.

PassTicketSTSModule
Validate mode:
- Validation key that the partner must use. Specify the keystore, keystore password, and a key or certificate to use.

Exchange mode:
- PassTicket key used to generate a valid PassTicket. This key value must consist of exactly 16 hexadecimal digits. This key is required *only* if you want to use a PassTicket token with the Web services security management component *and* you are *not* using z/OS®.
- Signing key used to sign outgoing tokens. Specify the keystore, keystore password, and a key or certificate to use.

X509STSModule
Validate mode:

The X.509 module has two configuration properties and both are related to keys and certificates:

X.509 validator identifier or class name
Specifies an alternative X.509 validator identifier or class name. When this value is not specified, the Tivoli Federated Identity Manager key service uses the default algorithm to validate the certificate path.

Enable X.509 certificate validation
Specifies whether validation of X.509 certificates must be enforced. By default, this check box is selected. When this box is cleared, the certificate is not validated. This option can be used in deployments where the certificate has already been validated by another entity.

Exchange mode: Exchange mode is not supported for X.509 tokens.

Chapter 6. Configuring your Web services application to use Web services security management

Configure deployment descriptors of the application for both generating and consuming tokens for your Web services application to use Web services security management.

Before you begin

The following sections show what the configuration values are for a SAML 2.0 token using the Rational Application Developer. Then, depending on what type of token you are using, you can change the values as necessary to apply to your specific token type.

See "Configuring WS-Security for the Echo application" on page 64, which shows sample values for Kerberos, SAML 1.1, Tivoli Access Manager, Username, and X.509 token types that were used in the Echo application.

About this task

You can also see the sample Echo client and Echo applications that are shipped with the Tivoli Federated Identity Manager product, as they also provide examples of this type of configuration.

Generating a SAML 2.0 token

About this task

This generation example generates a SAML 2.0 token from the contents of the logged in user's JAAS subject. It does not call the trust service.

Procedure

1. Use the following table to assist you in specifying the values on the Security Token panel.

Table 10. Values for generating a SAML 2.0 token on the Security Token panel

Field name	SAML 2.0 sample values
Name	SAML20
Local name	urn:oasis:names:tc:SAML:2.0:assertion#Assertion

2. Use the following table to assist you in specifying the values on the Token Generator panel.

Table 11. Values for generating a SAML 2.0 token on the Token Generator panel

Field name	SAML 2.0 sample values
Token generator name	SAML20
Token generator class	com.tivoli.am.fim.wssm.tokengenerators.WSSMTokenGenerator
Security token	SAML20
Use value type	Select check box
Value type	Custom Token

Field name	SAML 2.0 sample values
Local name	urn:oasis:names:tc:SAML:2.0:assertion#Assertion
Callback handler	com.tivoli.am.fim.wssm.callbackhandlers.WSSMTokenGeneratorCallbackHandler
Callback Handler Property Name (1)	xml.callback.handler.class.name
Callback Handler Property Value	com.tivoli.am.fim.wssm.callbackhandlers.JAASSubjectCallbackHandler
Callback Handler Property Name (2)	token.callback.handler.class.name
Callback Handler Property Value	com.tivoli.am.fim.wssm.callbackhandlers.SAMLACallbackHandler
Property Name (1)	sts.call
Property Value	false
Property Name (2)	issuer
Property Value	urn:itfim:wssm:tokengenerator

Consuming a SAML 2.0 token

About this task

This consumption example consumes a SAML 2.0 token by sending it to the trust service which exchanges it for a Username token. The Username token is used to perform the login at the Web service.

Procedure

1. Use the following table to assist you in specifying the values on the Required Security Token panel.

Table 12. Values for consuming a SAML 2.0 token on the Required Security Token panel

Field name	SAML 2.0 sample values
Name	SAML20
URI	urn:oasis:names:tc:SAML:2.0:assertion
Local name	Assertion
Usage type	Required

2. Use the following table to assist you in specifying the values on the Caller Part panel.

Table 13. Values for consuming a SAML 2.0 token on the Caller Part panel

Field name	SAML 2.0 sample values
Name	SAML20
Local name	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken
Property name (1)	com.ibm.wsspi.wssecurity.caller.tokenConsumerNS
Property value	urn:oasis:names:tc:SAML:2.0:assertion
Property name (2)	com.ibm.wsspi.wssecurity.caller.tokenConsumerLN
Property value	Assertion

3. Use the following table to assist you in specifying the values on the Token Consumer panel.

Table 14. Values for consuming a SAML 2.0 token on the Token Consumer panel

Field name	SAML 2.0 sample values
Token consumer name	SAML20
Token consumer class	com.tivoli.am.fim.wssm.tokensonsumers.WSSMTokenConsumer
Security token	SAML20
Use value type	Select check box
Value type	Custom Token
Local name	Assertion
URI	urn.oasis:names:tc:SAML:2.0:assertion
Use jaas.config	Select check box
jaas.config name	system:itfim.wssm.username

Chapter 7. Monitoring communications

You can monitor communications and activities within your Web services security management environment in several ways.

About this task

- Monitor the communications between the trust service and the server on which the Echo Web service application is running.
- Monitor the activities of the Web services security management component and Tivoli Federated Identity Manager by enabling logging.
- Monitor the communications between a Web service and a client. This procedure is explained in "Monitoring communications between the Echo client and the Echo application" on page 88.

Monitoring communications between the server and the trust service

You can use the TCPMonitor application utility to capture and monitor network communications between the server and the Tivoli Federated Identity Manager trust service. The Web service application and the trust service runs in the server.

Procedure

1. Start the TCPMonitor application provided with WebSphere Application Server. See the Tracing SOAP messages with tcpmon topic in the WebSphere Application Server Information Center for further information.
2. In the **Listen Port #** field, specify the same port number used for the trust service URL.
3. To configure the TCPMonitor application to act as a listener, ensure that **Listener** is selected, and then enter the following information:
   a. In the **Target Hostname** field, enter the host name of the WebSphere Application Server system. A host name of `localhost` can be used if the TCPMonitor application is running on the same system as WebSphere Application Server.
   b. In the **Target Port #** field, enter the port number that the server application is using for SOAP requests.
4. Click **Add** to configure the TCPMonitor application to listen for requests. After this operation completes, the fields in the panel are cleared and a new tab shows at the top of the window.
5. Click the **Port nnnnn** tab at the top of the window, where *nnnnn* is the port number that was added in the previous step.
6. Use the client application to perform an operation. The TCPMonitor window shows the request from the Web services security management component in the middle pane. It also shows the response from the trust service in the bottom pane.
7. Click **Switch Layout** in the TCPMonitor application, and resize the window if necessary, to show the request in the left pane and the response in the right pane.
8. Next, configure the settings for the trust service. Go to the directory where the wssm.properties file is located. The default directory is:

> **UNIX and Linux**
>
>   /opt/IBM/FIM/wssm/etc
>
> **Windows**
>
>   C:\Program Files\IBM\FIM\wssm\etc

9. Make a backup of the `wssm.properties` file.
10. Set the `sts.url` property to the port on which TCPMonitor is listening. For example: `sts.url=http://localhost:51962/TrustServer/SecurityTokenService`
11. Save the changes to the file.

What to do next

Note: The request and its associated data, and the response and its data, can be viewed by anyone with sufficient authority on the WebSphere Application Server system. It shows the exchange between one application server and another on the same system.

Enabling logging

You can enable logging for the Web services security management component and for Tivoli Federated Identity Manager.

Procedure

1. Start the WebSphere Application Server administrative console and log on, if necessary.

   Note: Ensure that you are using the administrative console associated with the application server where the Web services security management component is installed.
2. Click **Troubleshooting** > **Logs and Trace** to open the Logging and Tracing page.
3. Click the name of the server that you want to configure.
4. Click **Diagnostic Trace** and verify that diagnostic tracing is enabled. Diagnostic trace is enabled by default.
5. Click **Change Log Detail Levels** to open the Change Log Detail Levels page.
6. Click in the logging text box to edit the logging options text. This operation might take a minute or two to complete.
7. Enable logging for the Web services security management component and for Tivoli Federated Identity Manager by entering the following trace options in the text box all on one line and without any spaces:

   ```
   *=info
   :com.tivoli.am.fim.*=all
   :com.tivoli.am.fim.wssm.*=all
   ```

   Note: The options are shown on separate lines for readability in this document. The options must be entered all on one line without any spaces or tab or newline characters.
8. Scroll the window, if necessary, and click **Apply** to apply your changes.
9. In the Messages pane at the top of the Logging and Tracing window, click **Save** to commit your changes.
10. Restart the application server.

11. To view the log, locate the `trace.log` file associated with the application server. For example, if the application profile name is `BankApp` and the server name is `server1`, the default location for the `trace.log` file is:

AIX   `/usr/IBM/WebSphere/AppServer/profiles/BankApp/logs/server1`

UNIX and Linux
       `/opt/IBM/WebSphere/AppServer/profiles/BankApp/logs/server1`

Windows
       `C:\Program Files\IBM\WebSphere\AppServer\profiles\BankApp\logs\`
       `server1`

12. Open the file in a text editor to view it.

Chapter 8. Running the Echo demonstration application

Use the Echo demonstration application to validate that your WebSphere Application Server environment works correctly with Tivoli Federated Identity Manager. You can also use the Echo demonstration application to demonstrate the use of the Web services security management component.

Introduction to the Echo demonstration application

The Echo demonstration application is a Web service application for WebSphere Application Server that echoes text back to the Echo Web service client to illustrate the use of the Web services security management component.

The Echo demonstration application implements the Web service that is defined by the EchoService.wsdl file.

The file defines a number of ports to provide support for multiple token types, and to demonstrate token processing at both the Web service application, and the client.

The token types and their relative URLs, as defined in the WSDL file, are listed in the following table.

Table 15. Token types and associated relative URLs

Token types	Relative URL
No token	EchoApplication/services/ EchoServiceNoToken
Kerberos	EchoApplication/services/ EchoServiceKerberos
SAML 1.1	EchoApplication/services/ EchoServiceSAML11
SAML 2.0	EchoApplication/services/ EchoServiceSAML20
Tivoli Access Manager	EchoApplication/services/EchoServiceTAM
Username	EchoApplication/services/ EchoServiceUsername
X.509	EchoApplication/services/EchoServiceX509

Information about the WS-Security configuration of this application is provided in "Echo demonstration application" on page 18.

Configuring related components

Before installing and running the demonstration application, you must configure several components with information that is specific to the Echo demonstration application.

Before you begin

Note: Be sure you have completed the setup of your Web services security management component environment, including completing all tasks in the following chapters:

- Chapter 2, "Configuring WebSphere Application Server," on page 27.
- Chapter 3, "Updating configuration batch and script files," on page 39.

User registry considerations for the Echo application

The Echo Web service expects a login from a user named `wssm-testuser` with a password of `testonly123`. Therefore, you must add this user to the user registry or user registries that you are using for your Echo environment.

You can use any user registry that is supported by WebSphere Application Server. However, to ensure that the Echo application works correctly, consider the following tips:

- Set an environment where the Echo Web service application and Tivoli Access Manager use the same user registry. This setup simplifies your configuration because you must add the `wssm-testuser` to only that one registry.
- If you choose to use different registries for the Echo Web service and Tivoli Access Manager, you must add the `wssm-testuser` to both registries.

Note: To log in at the Echo client, you can use any user name and password that exists in the WebSphere Application Server user registry that is being used by the server where the Echo client is running.

Enabling Lightweight Third Party Authentication (LTPA) in WebSphere

You must enable Lightweight Third Party Authentication (LTPA) in WebSphere Application Server to run the demonstration application.

About this task

The following steps are only required if you are using WebSphere Application Server Version 6.0.2.x. LTPA is enabled by default in WebSphere Application Server Version 6.1.

Procedure

1. Start the WebSphere Application Server administrative console and log on, if necessary.

   Note: Ensure that you are using the administrative console associated with the application server where the Web services security management component is installed.

2. Click **Security** > **Global security**.
3. Open the **Authentication mechanisms** section, and click **LTPA**.
4. Enter a password in both the **Password** and **Confirm password** fields, and then click **Apply**.
5. In the Messages pane at the top of the window, click **Save** to commit your changes.

Configuring Tivoli Access Manager for Echo

The authorization check made during the execution of the Echo application module chain requires the creation and configuration of a Tivoli Access Manager object space. The Tivoli Access Manager object space must contain objects representing the Echo Web service application and its operations. Tivoli Access Manager users, groups, and ACLs must also be created.

Before you begin

The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

Creating protected objects for Echo

To create protected objects for the Echo Web service application, you can run the **wsdl2tam** command. This process creates a script file that you use with **pdadmin** to automate the creation of the space and objects.

Before you begin

The **wsdl2tam** script or batch file must be modified before performing this task. Both the Tivoli Access Manager policy server and authorization server must be running and the **pdadmin** command must be accessible in the command path.

Procedure

1. Open a command prompt on the system where WebSphere Application Server is installed.
2. Go to the directory that contains the files for the Web services security management component. The files can be found in the following default location:

   UNIX and Linux
   ```
   /opt/IBM/FIM/wssm/etc
   ```

   Windows
   ```
   C:\Program Files\IBM\FIM\wssm\etc
   ```
3. Run the modified **wsdl2tam** script or batch file specifying a unique logical name for the WSDL document (EchoWSDL), the name of the script file to be created (tamscript.txt), and the file name of the Echo WSDL file (EchoService.wsdl).

   UNIX and Linux
   ```
   ./wsdl2tam.sh -n EchoWSDL -tam tamscript.txt
     examples/etc/EchoService.wsdl
   ```

   > **Note:** When running this script on a Solaris system, you must use the **ksh** shell instead of the default shell (**/bin/sh**).

   Windows
   ```
   wsdl2tam.bat -n EchoWSDL -tam tamscript.txt
     C:\PROGRA~1\IBM\FIM\wssm\examples\etc\EchoService.wsdl
   ```
4. Open the script file and review its content to make sure that it is correct.
5. Save and close the file.
6. Open a command prompt on the system where Tivoli Access Manager is installed.

7. Next, use the **pdadmin** command to run the script file, such as `tamscript.txt` and create the protected object space.

   ```
   pdadmin -a sec_master -p testonly tamscript.txt
   ```

8. Optional: Verify that the protected object space was created using the **pdadmin** command or Web Portal Manager.

What to do next

Next, you must use the **pdadmin** to define users and groups. Then, create an ACL with permission [WebService]i and attach it to one of the objects in the WSSM object space.

Defining the access control policy for Echo

After you have created the object space for the Echo demonstration application, you must create users, groups, and the access control list.

Procedure

1. Open a command prompt on the system where Tivoli Access Manager is installed.
2. Use the **pdadmin** command to create an access control group called EchoUsers. EchoUsers must be able to start any Web service in the Echo demonstration application, and define a user called wssm-testuser. wssm-testuser is a member of that group and uses the password testonly123.

Example

```
pdadmin -a sec_master -p password
user create wssm-testuser cn=wssmtestuser,c=us wssm testuser testonly123
user modify wssm-testuser account-valid yes
group create EchoUsers cn=EchoUsers,c=us EchoUsers
group modify EchoUsers add wssm-testuser
acl create EchoUsers
acl modify EchoUsers set group EchoUsers T[WebService]i
acl attach /itfim-wssm/wssm-default/EchoWSDL EchoUsers
```

The two user commands create the user and activate the user account. The next two group commands create an access control group and assign the user to the group. Finally, the two acl commands create an access control list and attach that list to all of the access control objects associated with the Echo demonstration application. The result is that the wssm-testuser user ID has permission to start any EchoWSDL Web service.

What to do next

See the *IBM Tivoli Access Manager for e-business Administration Guide* for additional details on ACLs.

Configuring the X.509 key

If you use an X.509 token in the demonstration, you must configure the X.509 key using the key service in Tivoli Federated Identity Manager.

About this task

The X.509 token module needs an additional keystore file that is configured as a certificate authority so the certificate in the X.509 token can be validated. The X.509 token module contains the wssm_client certificate from the wssm_client.jks keystore file. This certificate was issued by the demonstration certificate authority

(itfim_demo_ca) that is part of Tivoli Federated Identity Manager. The certificate is stored in the `wssm_server.jks` keystore file, which is located, by default, in the following directory:

UNIX and Linux
/opt/IBM/FIM/wssm/examples/etc

Windows
C:\Program Files\IBM\FIM\wssm\examples\etc

The keystore file must be downloaded to your workstation, and then imported using the Tivoli Federated Identity Manager key service on the Tivoli Federated Identity Manager administration console.

Procedure

1. Start the Tivoli Federated Identity Manager administration console and log on, if necessary.
2. Click **Tivoli Federated Identity Manager** > **Key Service**. The Key Service portlet shows the keystores that are currently defined.
3. Click **Import.**
4. Complete the configuration information for the keystore file:
   a. In the **Location of the keystore file** field, specify the location on your workstation where you have downloaded the `wssm_server.jks` file.
   b. In the **Keystore Password** field, type `testonly`.
   c. In the **Keystore Name** field, type `wssm_server`.
   d. In the **Type list**, select **CA Certificates**.
5. Click **Finish**.

Configuring Kerberos for Echo

If you use a Kerberos STS module in the demonstration, you must follow these configuration instructions.

About this task

These steps are specific for use with Active Directory as your Key Distribution Center (KDC). If you use another type of KDC, modify these instructions as necessary.

Procedure

1. Make sure you have a DNS server installed.
2. Create a domain called, for example, `fim.test.com`.
3. Create a Kerberos realm called, for example, `FIM.TEST.COM`. The realm is automatically created for you, if you are using Active Directory.
4. Make sure you have the Windows 2003 SP2 support tools installed, which includes the Kerberos command line utilities.
5. Open **Administrative Tools** > **Active Directory Users and Computers** and create a user called `fimclient` with a password of `Passw0rd` under the domain `fim.test.com`. This user represents the Kerberos client.
6. Create a user called `fimservice` with a password of `Passw0rd` under `fim.test.com`. This user represents the Kerberos service.
7. Map the fimservice Kerberos principal name to the `fimservice` account using the following Windows **ksetup** command:

```
ksetup /mapuser fimservice/fimtest.fim.test.com@FIM.TEST.COM fimservice
```

where `fimtest.fim.test.com` is the fully-qualified domain name of the host machine on which the service is running.

8. Create a keytab file for `fimservice` using the following Windows **ktpass** command:

```
ktpass –out fimservice.keytab
   –princ fimservice/fimtest.fim.test.com@FIM.TEST.COM
   –Crypto RC4-HMAC-NT –mapUser fimservice@FIM.TEST.COM –pass Passw0rd
   –ptype KRB5_NT_PRINCIPAL
```

The location of the keytab file is specified in the Kerberos STS module's parameters for token validation panel in the console.

9. Modify the krb5.conf Kerberos configuration file that ships with Tivoli Federated Identity Manager:

```
[libdefaults]
    default_realm = FIM.TEST.COM
    default_tkt_enctypes = rc4-hmac
    default_tgs_enctypes = rc4-hmac

[realms]
    FIM.TEST.COM = {
        kdc = fimtest.fim.test.com:88
    }

[domain_realm]
    fim.test.com = FIM.TEST.COM
```

Ensure the kdc address is correct.

The location of the krb5.conf file is specified in the default Kerberos STS module initialization parameters panel in the console.

10. The jaas.conf Kerberos login configuration file is also part of the Tivoli Federated Identity Manager product. You do not need to modify this file. The location of the jaas.conf file is specified in the default Kerberos STS module initialization parameters panel in the console.

Configuring WS-Security for the Echo application

You must configure the deployment descriptors of the application to generate and consume tokens, so that your Web services application can use Web services security management.

In the Echo application, each token type has been configured. The following section shows the sample configuration values for SAML 1.1, Tivoli Access Manager, Username, and X.509 token types that were specified using the Rational Application Developer.

Note: Chapter 6, "Configuring your Web services application to use Web services security management," on page 51 shows the sample values for SAML 2.0.

Use this information as a guide when you are configuring your own Web services application to use Web services security management.

WS-Security configuration for Kerberos

Use the Rational Application Developer to define values for generating and consuming a Kerberos token.

Generating a Kerberos token
Procedure

1. Use the following table to assist you in specifying the values on the Security Token panel.

Table 16. Values for generating a Kerberos token on the Security Token panel

Field name	Kerberos sample values
Name	Kerberos
Local name	http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ

2. Use the following table to assist you in specifying the values on the Token Generator panel.

Table 17. Values for generating a Kerberos token on the Token Generator panel

Field name	Kerberos sample values
Token generator name	Kerberos
Token generator class	com.tivoli.am.fim.wssm.tokengenerators.WSSMTokenGenerator
Security token	Kerberos
Use value type	Select check box
Value type	Custom Token
Local name	http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ
Callback handler	com.tivoli.am.fim.wssm.callbackhandlers.WSSMTokenGeneratorCallbackHandler
Callback Handler Property Name (1)	xml.callback.handler.class.name
Callback Handler Property Value	com.tivoli.am.fim.wssm.callbackhandlers.JAASSubjectCallbackHandler
Callback Handler Property Name (2)	token.callback.handler.class.name
Callback Handler Property Value	com.tivoli.am.fim.wssm.callbackhandlers.KerberosCallbackHandler

Consuming a Kerberos token
Procedure

1. Use the following table to assist you in specifying the values on the Required Security Token panel.

Table 18. Values for consuming a Kerberos token on the Required Security Token panel

Field name	Kerberos sample values
Name	Kerberos
Local name	http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ
Usage type	Required

2. Use the following table to assist you in specifying the values on the Caller Part panel.

Table 19. Values for consuming a Kerberos token on the Caller Part panel

Field name	Kerberos sample values
Name	Kerberos
URI	urn:oasis:names:tc:SAML:2.0:assertion
Local name	Assertion
Property name (1)	com.ibm.wsspi.wssecurity.caller.tokenConsumerNS
Property value	(no value)
Property name (2)	com.ibm.wsspi.wssecurity.caller.tokenConsumerLN
Property value	http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ

3. Use the following table to assist you in specifying the values on the Token Consumer panel.

Table 20. Values for consuming a Kerberos token on the Token Consumer panel

Field name	Kerberos sample values
Token consumer name	Kerberos
Token consumer class	com.tivoli.am.fim.wssm.tokensonsumers.WSSMTokenConsumer
Security token	Kerberos
Use value type	Select check box
Value type	Custom Token
Local name	http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ
Use jaas.config	Select check box
jaas.config name	system:itfim.wssm.samla

WS-Security configuration for SAML 1.1

Use the Rational Application Developer to define values for generating and consuming a SAML 1.1 token.

Generating a SAML 1.1 token
Procedure

1. Use the following table to assist you in specifying the values on the Security Token panel.

Table 21. Values for generating a SAML 1.1 token on the Security Token panel

Field name	SAML 1.1 sample values
Name	SAML11
Local name	urn:oasis:names:tc:SAML:1.0:assertion#Assertion

2. Use the following table to assist you in specifying the values on the Token Generator panel.

Table 22. Values for generating a SAML 1.1 token on the Token Generator panel

Field name	SAML 1.1 sample values
Token generator name	SAML11

Table 22. Values for generating a SAML 1.1 token on the Token Generator panel (continued)

Field name	SAML 1.1 sample values
Token generator class	com.tivoli.am.fim.wssm.tokengenerators.WSSMTokenGenerator
Security token	SAML11
Use value type	Select check box
Value type	Custom Token
Local name	urn:oasis:names:tc:SAML:1.0:assertion#Assertion
Callback handler	com.tivoli.am.fim.wssm.callbackhandlers.WSSMTokenGeneratorCallbackHandler
Callback Handler Property Name (1)	xml.callback.handler.class.name
Callback Handler Property Value	com.tivoli.am.fim.wssm.callbackhandlers.JAASSubjectCallbackHandler
Callback Handler Property Name (2)	token.callback.handler.class.name
Callback Handler Property Value	com.tivoli.am.fim.wssm.callbackhandlers.SAMLACallbackHandler
Property Name (1)	sts.call
Property Value	false
Property Name (2)	issuer
Property Value	urn:itfim:wssm:tokengenerator

Consuming a SAML 1.1 token
Procedure

1. Use the following table to assist you in specifying the values on the Required Security Token panel.

Table 23. Values for consuming a SAML 1.1 token on the Required Security Token panel

Field name	SAML 1.1 sample values
Name	SAML11
URI	urn:oasis:names:tc:SAML:1.0:assertion
Local name	Assertion
Usage type	Required

2. Use the following table to assist you in specifying the values on the Caller Part panel.

Table 24. Values for consuming a SAML 1.1 token on the Caller Part panel

Field name	SAML 1.1 sample values
Name	SAML11
Local name	http://ibm.com/2004/01/itfim/ivcred
Property name (1)	com.ibm.wsspi.wssecurity.caller.tokenConsumerNS
Property value	urn:oasis:names:tc:SAML:1.0:assertion
Property name (2)	com.ibm.wsspi.wssecurity.caller.tokenConsumerLN
Property value	Assertion

3. Use the following table to assist you in specifying the values on the Token Consumer panel.

Table 25. Values for consuming a SAML 1.1 token on the Token Consumer panel

Field name	SAML 1.1 sample values
Token consumer name	SAML11
Token consumer class	com.tivoli.am.fim.wssm.tokensonsumers.WSSMTokenConsumer
Security token	SAML11
Use value type	Select check box
Value type	Custom Token
Local name	Assertion
URI	urn.oasis:names:tc:SAML:1.0:assertion
Use jaas.config	Select check box
jaas.config name	system:itfim.wssm.tam

WS-Security configuration for Tivoli Access Manager

Use the Rational Application Developer to define values for generating and consuming a Tivoli Access Manager token.

Before you begin

The information in this section applies to Tivoli Federated Identity Manager package users. It also applies to organizations that already have Tivoli Access Manager for e-business in their computing environment.

Generating a Tivoli Access Manager token
Procedure

1. Use the following table to assist you in specifying the values on the Security Token panel.

Table 26. Values for generating a Tivoli Access Manager token on the Security Token panel

Field name	Tivoli Access Manager sample values
Name	TAM
Local name	http://ibm.com/2004/01/itfim/ivcred

2. Use the following table to assist you in specifying the values on the Token Generator panel.

Table 27. Values for generating a Tivoli Access Manager token on the Token Generator panel

Field name	Tivoli Access Manager sample values
Token generator name	TAM
Token generator class	com.tivoli.am.fim.wssm.tokengenerators.WSSMTokenGenerator
Security token	TAM
Use value type	Select check box
Value type	Custom Token
Local name	http://ibm.com/2004/01/itfim/ivcred
Callback handler	am.fim.wssm.callbackhandlers.WSSMTokenGeneratorCallbackHandler

Table 27. Values for generating a Tivoli Access Manager token on the Token Generator panel (continued)

Field name	Tivoli Access Manager sample values
Callback Handler Property Name (1)	xml.callback.handler.class.name
Callback Handler Property Value	com.tivoli.am.fim.wssm.callbackhandlers.JAASSubjectCallbackHandler
Callback Handler Property Name (2)	token.callback.handler.class.name
Callback Handler Property Value	com.tivoli.am.fim.wssm.callbackhandlers.TAMCredentialCallbackHandler

Consuming a Tivoli Access Manager token
Procedure

1. Use the following table to assist you in specifying the values on the Required Security Token panel.

Table 28. Values for consuming a Tivoli Access Manager token on the Required Security Token panel

Field name	Tivoli Access Manager sample values
Name	TAM
Local name	http://ibm.com/2004/01/itfim/ivcred
Usage type	Required

2. Use the following table to assist you in specifying the values on the Caller Part panel.

Table 29. Values for consuming a Tivoli Access Manager token on the Caller Part panel

Field name	Tivoli Access Manager sample values
Name	TAM
URI	urn:oasis:names:tc:SAML:2.0:assertion
Local name	Assertion
Property name (1)	com.ibm.wsspi.wssecurity.caller.tokenConsumerNS
Property value	(no value)
Property name (2)	com.ibm.wsspi.wssecurity.caller.tokenConsumerLN
Property value	http://ibm.com/2004/01/itfim/ivcred

3. Use the following table to assist you in specifying the values on the Token Consumer panel.

Table 30. Values for the consuming a Tivoli Access Manager token on the Token Consumer panel

Field name	Tivoli Access Manager sample values
Token consumer name	TAM
Token consumer class	com.tivoli.am.fim.wssm.tokensonsumers.WSSMTokenConsumer
Security token	TAM
Use value type	Select check box
Value type	Custom Token
Local name	http://ibm.com/2004/01/itfim/ivcred
Use jaas.config	Select check box

Table 30. Values for the consuming a Tivoli Access Manager token on the Token Consumer panel (continued)

Field name	Tivoli Access Manager sample values
jaas.config name	system:itfim.wssm.samla

WS-Security configuration for Username

Use the Rational Application Developer to define values for generating and consuming a Username token.

Generating a Username token
Procedure

1. Use the following table to assist you in specifying the values on the Security Token panel.

Table 31. Values for generating a Username token on the Security Token panel

Field name	Username sample values
Name	Username
Local name	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken

2. Use the following table to assist you in specifying the values on the Token Generator panel.

Table 32. Values for generating a Username token on the Token Generator panel

Field name	Username sample values
Token generator name	Username
Token generator class	com.tivoli.am.fim.wssm.tokengenerators.WSSMTokenGenerator
Security token	Username
Use value type	Select check box
Value type	Username Token
Local name	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken
Callback handler	com.tivoli.am.fim.wssm.callbackhandlers.WSSMTokenGeneratorCallbackHandler
Callback Handler Property Name (1)	xml.callback.handler.class.name
Callback Handler Property Value	com.tivoli.am.fim.wssm.callbackhandlers.JAASSubjectCallbackHandler
Callback Handler Property Name (2)	token.callback.handler.class.name
Callback Handler Property Value	com.tivoli.am.fim.wssm.callbackhandlers.UsernameCallbackHandler

Consuming a Username token
Procedure

1. Use the following table to assist you in specifying the values on the Required Security Token panel.

Table 33. Values for consuming a Username token on the Required Security Token panel

Field name	Username sample values
Name	Username
Local name	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken
Usage type	Required

   2. Use the following table to assist you in specifying the values on the Caller Part panel.

Table 34. Values for consuming a Username token on the Caller Part panel

Field name	Username sample values
Name	Username
URI	urn:oasis:names:tc:SAML:2.0:assertion
Local name	Assertion
Property name (1)	com.ibm.wsspi.wssecurity.caller.tokenConsumerNS
Property value	(no value)
Property name (2)	com.ibm.wsspi.wssecurity.caller.tokenConsumerLN
Property value	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken

   3. Use the following table to assist you in specifying the values on the Token Consumer panel.

Table 35. Values for consuming a Username token on the Token Consumer panel

Field name	Username sample values
Token consumer name	Username
Token consumer class	com.tivoli.am.fim.wssm.tokensonsumers.WSSMTokenConsumer
Security token	Username
Use value type	Select check box
Value type	Username Token
Local name	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken
Use jaas.config	Select check box
jaas.config name	system:itfim.wssm.samla

WS-Security configuration for X.509

Use the Rational Application Developer to define values for generating and consuming an X.509 token.

Generating an X.509 token
Procedure

   1. Use the following table to assist you in specifying the values on the Security Token panel.

Table 36. Values for generating an X.509 token on the Security Token panel

Field name	X.509 sample values
Name	X509
Local name	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509

2. Use the following table to assist you in specifying the values on the Token Generator panel.

Table 37. Values for generating an X.509 token on the Token Generator panel

Field name	X.509 sample values
Token generator name	X509
Token generator class	com.ibm.wsspi.wssecurity.token.X509TokenGenerator
Security token	X509
Use value type	Select check box
Value type	X509 certificate token
Local name	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509
Callback handler	com.ibm.wsspi.wssecurity.auth.callback.X509CallbackHandler
User key store	Select check box
Password	testonly
Path	${ITFIM_WSSM}/examples/etc/wssm_client.jks
Type	JKS
Key Alias	wssm_client
Key password	testonly
Key name	cn=wssm_client,ou=Tivoli,o=ibm,c=US

Consuming an X.509 token
Procedure

1. Use the following table to assist you in specifying the values on the Required Security Token panel.

Table 38. Values for consuming an X.509 token on the Required Security Token panel

Field name	X.509 sample values
Name	X509
Local name	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509
Usage type	Required

2. Use the following table to assist you in specifying the values on the Caller Part panel.

Table 39. Values for consuming an X.509 token on the Caller Part panel

Field name	X.509 sample values
Name	X509
URI	urn:oasis:names:tc:SAML:1.0:assertion
Local name	Assertion
Property name (1)	com.ibm.wsspi.wssecurity.caller.tokenConsumerNS

Field name	X.509 sample values
Property value	(no value)
Property name (2)	com.ibm.wsspi.wssecurity.caller.tokenConsumerLN
Property value	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509

3. Use the following table to assist you in specifying the values on the Token Consumer panel.

Table 40. Values for consuming an X.509 token on the Token Consumer panel

Field name	X.509 sample values
Token consumer name	X509
Token consumer class	com.tivoli.am.fim.wssm.tokensonsumers.WSSMTokenConsumer
Security token	X509
Use value type	Select check box
Value type	X509 certificate token
Local name	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509
Use jaas.config	Select check box
jaas.config name	system:itfim.wssm.samla

Installing the Echo Web service application

Before you can run the Echo demonstration application, you must install the Echo Web service application from the WebSphere Application Server administrative console.

Procedure

1. Start the WebSphere Application Server administrative console and log on, if necessary.

   Note: Ensure that you are using the administrative console associated with the application server where the Web services security management component is installed.

2. Click **Applications** > **Install New Application** to open the Preparing for the application installation page.

3. Click **Browse** and locate the EAR file, `echoapplication.ear`, containing the Echo Web service application. The binary files are located in the `wssm/examples/bin` directory.

4. Click **Next** on the remaining configuration panels to accept the default values.

5. Click **Finish** to install the application.

6. After the secured Echo application has been installed, click **Save to Master Configuration** to save your changes.

7. Optional: You can confirm that the application is running by opening a Web browser and entering the URL for the location of the application. For example,

the URL might be `http://localhost:9080/EchoApplication/services/`
`EchoServiceNoToken`. If the application was installed successfully a "Hi there"
message shows.

Configuring the Echo Web service application

Before you can run the demonstration, you must configure the Echo Web service
application by creating and configuring the application module chain and
configuring the partner chains for the token types you want to demonstrate.

Configuring Echo application STS module chains

To use the Echo demonstration application, you must configure application STS
module chains.

About this task

These chains are executed when the WSSM token consumer calls the STS to
validate the token received in the Web request by the Echo application and to issue
a token for login. Each chain shares the same mapping rules defined in the
supplied wssm_test.xsl. The default location of this file is `C:\Program`
`Files\IBM\FIM\wssm\examples\etc\wssm_test.xsl`.

Use the instructions below to create the following trust service chains:
- EchoApplication Kerberos to SAML 2.0
- EchoApplication SAML 1.1 to TAM
- EchoApplication SAML 2.0 to Username
- EchoApplication TAM to SAML 2.0
- EchoApplication Username to SAML 2.0
- EchoApplication X.509 to SAML 1.1

You can load the configuration changes to the runtime after creating each of these
chains, or you can load them once after creating all of the chains.

Procedure
1. Create an STS module chain to validate a Kerberos token and issue a SAML 2.0
   token. Use the Trust Service Chain Mapping wizard in the console and specify
   the following properties for this chain:
   - On the Chain Mapping Identification panel, set:

Chain Mapping Name	EchoApplication Kerberos to SAML 2.0

   - On the Chain Mapping Lookup panel, set:

AppliesTo Address	REGEXP:(.*/EchoApplication/services/ EchoServiceKerberos)
Issuer Address	urn:itfim:wssm:tokenconsumer

   - On the Chain Assembly panel, add the following module instances in this
     order:

Order	Module Instance Name	Mode
1	Default Kerberos Token	validate

Order	Module Instance Name	Mode
2	Default Map Module	map
3	Default TAM Authorization	authorize
4	Default SAML 2.0 Token	issue

Click **Continue** in the message box that says that the module chain you have assembled does not meet the required Trust Service module chain structure.

- On the Kerberos Module configuration panel, set:

Kerberos service keytab file name	C:\Program Files\IBM\FIM\wssm\examples\etc\fimservice.keytab

- On the Default Map Module configuration panel, set:

XSLT File Containing Identity Mapping Rule	wssm_test.xsl

- On the Tivoli Access Manager Authorization Module configuration panel, use the default values.

  Note: The wssm_test.xsl mapping rules specify the Tivoli Access Manager authorization object name has a prefix of `/itfim-wssm/wssm-default/EchoWSDL/EchoService/EchoService`. This name must match the name in the Tivoli Access Manager object space.

- On the SAML 2.0 Module Configuration panel, set:

The name of the organization issuing the assertions	fim.test.com
Sign SAML Assertions	Clear the check box

Use the defaults for the fields that are not specified above.

2. Create an STS module chain to validate a SAML 1.1 token and issue a Tivoli Access Manager token. The module chain also maps the remote identity to a local identity and performs a Tivoli Access Manager authorization check to ensure the caller is authorized to invoke the Web service operation.

   Use the Trust Service Chain Mapping wizard in the console and specify the following properties for this chain:

   - On the Chain Mapping Identification panel, set:

Chain Mapping Name	EchoApplication SAML 1.1 to TAM

   - On the Chain Mapping Lookup panel, set:

AppliesTo Address	REGEXP:(.*/EchoApplication/services/EchoServiceSAML11)
Issuer Address	urn:itfim:wssm:tokenconsumer

   - On the Chain Assembly panel, add the following module instances in this order:

Order	Module Instance Name	Mode
1	Default SAML 1.1 Token	validate
2	Default Map Module	map

Order	Module Instance Name	Mode
3	Default TAM Authorization	authorize
4	Default IVCred Token	issue

Click **Continue** in the message box that says that the module chain you have assembled does not meet the required Trust Service module chain structure.

- On the SAML 1.1 Module Configuration panel, set:

Enable one-time assertion use enforcement	Clear the check box.
Enable Signature Validation	Clear the check box.

- On the Default Map Module configuration panel, set:

XSLT File Containing Identity Mapping Rule	wssm_test.xsl

- On the Tivoli Access Manager Authorization Module configuration panel, use the default values.

  Note: The wssm_test.xsl mapping rules specify the Tivoli Access Manager authorization object name has a prefix of /itfim-wssm/wssm-default/ EchoWSDL/EchoService/EchoService. This name must match the name in the Tivoli Access Manager object space.

- On the IVCred Module configuration panel, set:

Enable signatures	Select the check box.
Keystore Password	Type the password and select the key from the list. The default password is testonly.

Use the defaults for the fields that are not specified above.

3. Create an STS module chain to validate a SAML 2.0 token and issue a Username token. The module chain also maps the remote identity to a local identity and performs a Tivoli Access Manager authorization check to ensure the caller is authorized to invoke the Web service operation.

   Use the Trust Service Chain Mapping wizard in the console and specify the following properties for this chain:

   - On the Chain Mapping Identification panel, set:

Chain Mapping Name	EchoApplication SAML 2.0 to Username

   - On the Chain Mapping Lookup panel, set:

AppliesTo Address	REGEXP:(.*/EchoApplication/services/ EchoServiceSAML20)
Issuer Address	urn:itfim:wssm:tokenconsumer

   - On the Chain Assembly panel, add the following module instances in this order:

Order	Module Instance Name	Mode
1	Default SAML 2.0 Token	validate
2	Default Map Module	map

Order	Module Instance Name	Mode
3	Default TAM Authorization	authorize
4	Default Username Token	issue

Click **Continue** in the message box that says that the module chain you have assembled does not meet the required Trust Service module chain structure.

- On the SAML 2.0 Module Configuration panel, set:

Enable one-time assertion use enforcement	Clear the check box.
Enable Signature Validation	Clear the check box.

- On the Default Map Module configuration panel, set:

XSLT File Containing Identity Mapping Rule	`wssm_test.xsl`

- On the Tivoli Access Manager Authorization Module configuration panel, use the default values.

  Note: The wssm_test.xsl mapping rules specify the Tivoli Access Manager authorization object name has a prefix of `/itfim-wssm/wssm-default/ EchoWSDL/EchoService/EchoService`. This name must match the name in the Tivoli Access Manager object space.

- On the Username Token Module configuration panel, use the default values.

  Use the defaults for the fields that are not specified above.

4. Create an STS module chain to validate a Tivoli Access Manager token and issue a SAML 2.0 token. The module chain also maps the remote identity to a local identity and performs a Tivoli Access Manager authorization check to ensure the caller is authorized to invoke the Web service operation.

  Use the Trust Service Chain Mapping wizard in the console and specify the following properties for this chain:

  - On the Chain Mapping Identification panel, set:

Chain Mapping Name	`EchoApplication TAM to SAML 2.0`

  - On the Chain Mapping Lookup panel, set:

AppliesTo Address	`REGEXP:(.*/EchoApplication/services/ EchoServiceTAM)`
Issuer Address	`urn:itfim:wssm:tokenconsumer`

  - On the Chain Assembly panel, add the following module instances in this order:

Order	Module Instance Name	Mode
1	Default IVCred Token	validate
2	Default Map Module	map
3	Default TAM Authorization	authorize
4	Default SAML 2.0 Token	issue

Click **Continue** in the message box that says that the module chain you have assembled does not meet the required Trust Service module chain structure.

- On the IVCred Module configuration panel, set:

Enable signatures	Select the check box.
Keystore Password	Type the password and select the key from the list.

- On the Default Map Module configuration panel, set:

XSLT File Containing Identity Mapping Rule	wssm_test.xsl

- On the Tivoli Access Manager Authorization Module configuration panel, use the default values.

  Note: The wssm_test.xsl mapping rules specify the Tivoli Access Manager authorization object name has a prefix of /itfim-wssm/wssm-default/ EchoWSDL/EchoService/EchoService. This name must match the name in the Tivoli Access Manager object space.
- On the SAML 2.0 Token Module configuration panel, set:

The name of the organization issuing the assertions	test
Sign SAML Assertions	Clear the check box.

Use the defaults for the fields that are not specified above.

5. Create an STS module chain to validate a Username token and issue a SAML 2.0 token. The module chain also maps the remote identity to a local identity and performs a Tivoli Access Manager authorization check to ensure the caller is authorized to invoke the Web service operation.

Use the Trust Service Chain Mapping wizard in the console and specify the following properties for this chain:

- On the Chain Mapping Identification panel, set:

Chain Mapping Name	EchoApplication Username to SAML 2.0

- On the Chain Mapping Lookup panel, set:

AppliesTo Address	REGEXP:(.*/EchoApplication/services/ EchoServiceUsername)
Issuer Address	urn:itfim:wssm:tokenconsumer

- On the Chain Assembly panel, add the following module instances in this order:

Order	Module Instance Name	Mode
1	Default Username Token	validate
2	Default TAM Authentication	authenticate
3	Default Map Token	map
4	Default TAM Authorization	authorize
5	Default SAML 2.0 Token	issue

Click **Continue** in the message box that says that the module chain you have assembled does not meet the required Trust Service module chain structure.

- On the Username Module Configuration panel, set:

Skip password validation	Select the check box.

- On the Tivoli Access Manager Authentication Module configuration panel, there are no values to set.
- On the Default Map Module configuration panel, set:

XSLT File Containing Identity Mapping Rule	wssm_test.xsl

- On the Tivoli Access Manager Authorization Module configuration panel, use the default values.

  Note: The wssm_test.xsl mapping rules specify the Tivoli Access Manager authorization object name has a prefix of /itfim-wssm/wssm-default/ EchoWSDL/EchoService/EchoService. This name must match the name in the Tivoli Access Manager object space.

- On the SAML 2.0 Token Module configuration panel, set:

The name of the organization issuing the assertions	test
Sign SAML Assertions	Clear the check box.

  Use the defaults for the fields that are not specified above.
6. Create an STS module chain to validate an X.509 token and issue a SAML 1.1 token. The module chain also maps the remote identity to a local identity and performs a Tivoli Access Manager authorization check to ensure the caller is authorized to invoke the Web service operation.

   Use the Trust Service Chain Mapping wizard in the console and specify the following properties for this chain:
   - On the Chain Mapping Identification panel, set:

Chain Mapping Name	EchoApplication X.509 to SAML 1.1

   - On the Chain Mapping Lookup panel, set:

AppliesTo Address	REGEXP:(.*/EchoApplication/services/ EchoServiceX509)
Issuer Address	urn:itfim:wssm:tokenconsumer

   - On the Chain Assembly panel, add the following module instances in this order:

Order	Module Instance Name	Mode
1	Default X.509 Token	validate
2	Default Map Token	map
3	Default TAM Authorization	authorize
4	Default SAML 1.1 Token	issue

   Click **Continue** in the message box that says that the module chain you have assembled does not meet the required Trust Service module chain structure.
   - On the X.509 Module Configuration panel, use the default values.

- On the Default Map Module configuration panel, set:

XSLT File Containing Identity Mapping Rule	wssm_test.xsl

- On the Tivoli Access Manager Authorization Module configuration panel, use the default values.

  Note: The wssm_test.xsl mapping rules specify the Tivoli Access Manager authorization object name has a prefix of `/itfim-wssm/wssm-default/ EchoWSDL/EchoService/EchoService`. This name must match the name in the Tivoli Access Manager object space.

- On the SAML 1.1 Token Module configuration panel, set:

The name of the organization issuing the assertions	test
Sign SAML Assertions	Clear the check box.

Use the defaults for the fields that are not specified above.

Configuring dynamic Echo application STS module chains

You can use dynamic chains to configure the Echo application.

Before you begin

A description of dynamic chains is available in the *IBM Tivoli Federated Identity Manager Administration Guide*.

Before you begin, be sure to delete any conflicting module chains. For example, if there is already a module chain that validates a Tivoli Access Manager token for the echo application, this chain must be deleted before creating the equivalent dynamic chain.

About this task

As the issuing of a SAML 2.0 token may be performed independently of the incoming Tivoli Access Manager or Username token being validated, a dynamic chain is used. The validating of the Tivoli Access Manager and Username token is performed in separate chains.

You can load the configuration changes to the runtime after creating each of these chains, or you can load them once after creating all of the chains.

Procedure

1. Create an STS module chain to validate a Tivoli Access Manager token. Use the Trust Service Chain Mapping wizard in the console and specify the following properties for this chain:

   - On the Chain Mapping Identification panel, set:

Chain Mapping Name	EchoApplication validate TAM

   - On the Chain Mapping Lookup panel, set:

AppliesTo Address	REGEXP:(.*/EchoApplication/services/ EchoServiceTAM)
Issuer Address	urn:itfim:wssm:tokenconsumer

- On the Chain Assembly panel, add the following module instances in this order:

Order	Module Instance Name	Mode
1	Default IVCred Token	validate
2	Default Map Module	map
3	Default Dynamic Chain Module Instance	other

Click **Continue** in the message box that says that the module chain you have assembled does not meet the required Trust Service module chain structure.
- On the IVCred Module configuration panel, use the default settings.
- On the Default Map Module configuration panel, set:

XSLT File Containing Identity Mapping Rule	wssm_test.xsl

- The Dynamic Chain Module configuration panel has no properties to set.

Use the defaults for the fields that are not specified above.

2. Create an STS module chain to validate a Username token. Use the Trust Service Chain Mapping wizard in the console and specify the following properties for this chain:
   - On the Chain Mapping Identification panel, set:

Chain Mapping Name	EchoApplication validate Username

- On the Chain Mapping Lookup panel, set:

AppliesTo Address	REGEXP:(.*/EchoApplication/services/ EchoServiceUsername)
Issuer Address	urn:itfim:wssm:tokenconsumer

- On the Chain Assembly panel, add the following module instances in this order:

Order	Module Instance Name	Mode
1	Default Username Token	validate
2	Default Map Module	map
3	Default Dynamic Chain Module Instance	other

Click **Continue** in the message box that says that the module chain you have assembled does not meet the required Trust Service module chain structure.
- On the Username Token Module Configuration panel, set:

Skip password validation	Select the check box.

- On the Default Map Module configuration panel, set:

XSLT File Containing Identity Mapping Rule	wssm_test.xsl

- The Dynamic Chain Module configuration panel has no properties to set.

Use the defaults for the fields that are not specified above.

3. Create a dynamic STS module chain to issue a SAML 2.0 token. Use the Trust Service Chain Mapping wizard in the console and specify the following properties for this chain:

- On the Chain Mapping Identification panel, set:

Chain Mapping Name	EchoApplication issue SAML 2.0 (dynamic)
Create a dynamic chain	Select the check box.

- On the Chain Mapping Lookup panel, set:

AppliesTo Address	REGEXP:(.*/EchoApplication/services/ EchoService.*)
Issuer Address	urn:itfim:wssm:tokenconsumer

- On the Chain Assembly panel, add the following module instances in this order:

Order	Module Instance Name	Mode
1	Default SAML 2.0 Token	issue

Click **Continue** in the message box that says that the module chain you have assembled does not meet the required Trust Service module chain structure.

- On the SAML 2.0 Module Configuration panel, set:

The name of the organization issuing the assertions	fim.test.com
Sign SAML Assertions	Clear the check box.

Use the defaults for the fields that are not specified above.

Installing the Echo Client

Install the client for the Echo demonstration application.

Procedure

1. Start the WebSphere Application Server administrative console and log on, if necessary.

   Note: Ensure that you are using the administrative console associated with the application server where the Web services security management component is installed.

2. Click **Applications** > **Install New Application** to open the Preparing for the application installation page.

3. Click **Browse** and locate the EAR file, echoclientapplication.ear, containing the Echo Web service client. The binary files are located in the wssm/examples/bin directory.

4. Click **Next** in the remaining configuration panels, accepting the default values.

5. Click **Finish** to install the application.

6. After the secured Echo application has been installed, click **Save to Master Configuration** to save your changes.
7. Optional: You can confirm that the application is running by opening a Web browser and entering the URL for the location of the application. For example, the URL might be `http://localhost:9080/EchoClientApplication/default.jsp`.

   If the application was installed successfully, you are prompted for a user name and password. This user name and password must be configured in the WebSphere Application Server user registry. After you successfully log in, an Echo page displays.

Configuring the Echo Client

Configure the Echo Web service client by creating and configuring the application module chain, and the partner chain.

Configuring Echo client application STS module chains

To use the Echo demonstration application, you must configure client application STS module chains.

About this task

These chains are executed when the WSSM token generator calls the STS to generate a token for inclusion in the Web service request being sent by the Echo client application.

Use the instructions below to create the following trust service chains:
- EchoClientApplication SAML 2.0 to Kerberos
- EchoClientApplication SAML 2.0 to TAM
- EchoClientApplication SAML 2.0 to Username

You can load the configuration changes to the runtime after creating each of these chains, or you can load them once after creating all of the chains.

Procedure
1. Create an STS module chain to validate a SAML 2.0 token and issue a Kerberos token. Use the Trust Service Chain Mapping wizard in the console and specify the following properties for this chain:
   - On the Chain Mapping Identification panel, set:

Chain Mapping Name	EchoClientApplication SAML 2.0 to Kerberos

   - On the Chain Mapping Lookup panel, set:

AppliesTo Address	REGEXP:(.*/EchoApplication/services/ EchoServiceKerberos)
Issuer Address	urn:itfim:wssm:tokengenerator

   - On the Chain Assembly panel, add the following module instances in this order:

Order	Module Instance Name	Mode
1	Default SAML 2.0 Token	validate

Order	Module Instance Name	Mode
2	Default Map Module	map
3	Default Kerberos Token	issue

Click **Continue** in the message box that says that the module chain you have assembled does not meet the required Trust Service module chain structure.

- On the SAML 2.0 Module Configuration panel, set:

Enable one-time assertion use enforcement	Clear the check box.
Enable Signature Validation	Clear the check box.

- On the Default Map Module configuration panel, set:

XSLT File Containing Identity Mapping Rule	wssm_test.xsl

- On the Kerberos Module configuration panel, set:

Kerberos realm name	FIM.TEST.COM
Kerberos service name	fimservice/fimtest.fim.test.com

Use the default value for the fields that are not specified in the preceding information.

2. Create an STS module chain to validate a SAML 2.0 token and issue a Tivoli Access Manager token. Use the Trust Service Chain Mapping wizard in the console and specify the following properties for this chain:

- On the Chain Mapping Identification panel, set:

Chain Mapping Name	EchoClientApplication SAML 2.0 to TAM

- On the Chain Mapping Lookup panel, set:

AppliesTo Address	REGEXP:(.*/EchoApplication/services/ EchoServiceTAM)
Issuer Address	urn:itfim:wssm:tokengenerator

- On the Chain Assembly panel, add the following module instances in this order:

Order	Module Instance Name	Mode
1	Default SAML 2.0 Token	validate
2	Default IVCred Token	issue

Click **Continue** in the message box that says that the module chain you have assembled does not meet the required Trust Service module chain structure.

- On the SAML 2.0 Module Configuration panel, set:

Enable one-time assertion use enforcement	Clear the check box.
Enable Signature Validation	Clear the check box.

- On the IVCred Module configuration panel, set:

Enable signatures	Select the check box.

Keystore Password	Type the password and select the key from the list. The default password is `testonly`.

Use the default value for the fields that are not specified in the preceding information.

3. Create an STS module chain to validate a SAML 2.0 token and issue a Username token. Use the Trust Service Chain Mapping wizard in the console and specify the following properties for this chain:

   • On the Chain Mapping Identification panel, set:

Chain Mapping Name	`EchoClientApplication SAML 2.0 to Username`

   • On the Chain Mapping Lookup panel, set:

AppliesTo Address	`REGEXP:(.*/EchoApplication/services/EchoServiceUsername)`
Issuer Address	`urn:itfim:wssm:tokengenerator`

   • On the Chain Assembly panel, add the following module instances in this order:

Order	Module Instance Name	Mode
1	Default SAML 2.0 Token	validate
2	Default Username Token	issue

Click **Continue** in the message box that says that the module chain you have assembled does not meet the required Trust Service module chain structure.

   • On the SAML 2.0 Module Configuration panel, set:

Enable one-time assertion use enforcement	Clear the check box.
Enable Signature Validation	Clear the check box.

   • On the Username Module configuration panel, use the default values.

Use the default value for the fields that are not specified in the preceding information.

Configuring dynamic Echo client application STS module chains

You can use dynamic chains to configure the Echo application.

Before you begin

A description of dynamic chains is available in the *IBM Tivoli Federated Identity Manager Administration Guide*.

Before you begin, be sure to delete any conflicting module chains. For example, if there is already a module chain that issues a Tivoli Access Manager token for the echo client application, this chain must be deleted before creating the equivalent dynamic chain.

About this task

As the issuing of Tivoli Access Manager and Username tokens both involve first
validating a SAML 2.0 token, this common functionality may be contained within a
single chain. The actual issuing of the Tivoli Access Manager or Username token is
performed in separate dynamic chains.

You can load the configuration changes to the runtime after creating each of these
chains, or you can load them once after creating all of the chains.

Procedure

1. Create an STS module chain to validate a SAML 2.0 token. Use the Trust
   Service Chain Mapping wizard in the console and specify the following
   properties for this chain:
   - On the Chain Mapping Identification panel, set:

Chain Mapping Name	`EchoClientApplication validate SAML 2.0`

   - On the Chain Mapping Lookup panel, set:

AppliesTo Address	`REGEXP:(.*/EchoApplication/services/` `EchoService.*)`
Issuer Address	`urn:itfim:wssm:tokengenerator`

   - On the Chain Assembly panel, add the following module instances in this
     order:

Order	Module Instance Name	Mode
1	Default SAML 2.0 Token	validate
2	Default Map Module	map
3	Default Dynamic Chain Module Instance	other

   Click **Continue** in the message box that says that the module chain you have
   assembled does not meet the required Trust Service module chain structure.
   - On the SAML 2.0 Module Configuration panel, set:

Enable Signature Validation	Clear the check box.

   - On the Default Map Module configuration panel, set:

XSLT File Containing Identity Mapping **Rule**	`wssm_test.xsl`

   - The Dynamic Chain Module configuration panel has no properties to set.

   Use the default value for the fields that are not specified in the preceding
   information.
2. Create a dynamic STS module chain to issue a Tivoli Access Manager token.
   Use the Trust Service Chain Mapping wizard in the console and specify the
   following properties for this chain:
   - On the Chain Mapping Identification panel, set:

Chain Mapping Name	`EchoClientApplication issue TAM` `(dynamic)`

Create a dynamic chain	Select the check box.

- On the Chain Mapping Lookup panel, set:

AppliesTo Address	REGEXP:(.*/EchoApplication/services/EchoServiceTAM)
Issuer Address	urn:itfim:wssm:tokengenerator

- On the Chain Assembly panel, add the following module instances in this order:

Order	Module Instance Name	Mode
1	Default IVCred Token	issue

Click **Continue** in the message box that says that the module chain you have assembled does not meet the required Trust Service module chain structure.

- On the IVCred Module configuration panel, use the default settings.

Use the default value for the fields that are not specified in the preceding information.

3. Create a dynamic STS module chain to issue a Username token. Use the Trust Service Chain Mapping wizard in the console and specify the following properties for this chain:

- On the Chain Mapping Identification panel, set:

Chain Mapping Name	EchoClientApplication issue Username (dynamic)
Create a dynamic chain	Select the check box.

- On the Chain Mapping Lookup panel, set:

AppliesTo Address	REGEXP:(.*/EchoApplication/services/EchoServiceUsername)
Issuer Address	urn:itfim:wssm:tokengenerator

- On the Chain Assembly panel, add the following module instances in this order:

Order	Module Instance Name	Mode
1	Default Username Token	issue

Click **Continue** in the message box that says that the module chain you have assembled does not meet the required Trust Service module chain structure.

- On the Username Module configuration panel, use the default values.

Use the default value for the fields that are not specified in the preceding information.

Monitoring the Echo service and client

You can monitor the Echo demonstration application in several ways.

About this task

- Monitor the communications between the trust service and the server on which the Echo Web service application is running. See "Monitoring communications between the server and the trust service" on page 55 for details.
- Monitor the activities of the Web services security management component and Tivoli Federated Identity Manager by enabling logging. See "Enabling logging" on page 56 for details.
- Monitor the communications between the Echo service and the Echo client, as described in "Monitoring communications between the Echo client and the Echo application."
- Monitor the progress of the WebSphere Application Server, as described in "Monitoring progress" on page 89.

Monitoring communications between the Echo client and the Echo application

You can use the TCPMonitor application utility to capture and monitor network communications between the Echo client and the Echo Web service application. The Echo application runs in a server.

Before you begin

Ensure that the server used by the Echo Web service application is running.

Procedure

1. Start the TCPMonitor application provided with WebSphere Application Server. See the topic Tracing SOAP messages with tcpmon in the WebSphere Application Server Information Center for more information.
2. In the **Listen Port #** field, specify a port number to use for incoming requests. Select a port number that is not currently in use on the system.
3. To configure the TCPMonitor application to act as a listener, select **Listener**, and then enter the following information:
   a. In the **Target Hostname** field, enter the host name of the WebSphere Application Server system. A host name of `localhost` can be used if the TCPMonitor application is running on the same system as WebSphere Application Server.
   b. In the **Target Port #** field, enter the port number that the server application is using for SOAP requests.
4. Click **Add** to configure the TCPMonitor application to listen for requests. For example, requests received on port 51962 might be sent to port 9080. Conversely, information returned from port 9080 is forwarded back to the client on port 51962. After this operation completes, the fields in the panel are cleared and a new tab shows at the top of the window.
5. Click the **Port nnnnn** tab at the top of the window, where *nnnnn* is the port number that was added in the previous step. After selecting the tab, the monitor panel shows
6. Use the Echo client to perform an operation. The TCPMonitor window shows the request in the middle pane and the response in the bottom pane.
7. Click **Switch Layout** in the TCPMonitor application. Resize the window if necessary, to show the request on the left pane and the response in the right pane.

Monitoring progress

You can use the WebSphere Application Server log files to monitor its progress.

About this task

See the WebSphere Application Server information center at http://
publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp for more information
about how to configure the available log files.

Starting the Echo demonstration application

You can start the Echo demonstration application from the WebSphere Application
Server administrative console.

Procedure

1. Start the WebSphere Application Server administrative console and log on, if
   necessary.

   Note: Ensure that you are using the administrative console associated with the
   application server where the Web services security management component is
   installed.
2. Click **Applications** > **Enterprise Applications** to open the Enterprise
   Applications page.
3. Select the EchoApplication server application, and click **Start**.

Running the Echo client

Start the Echo client. The client interacts with the Echo demonstration application.

Before you begin

Before beginning, ensure that the user profile you use for login is configured in the
WebSphere Application Server user registry.

Procedure

1. Open a Web browser and go to the location where the Echo client is installed,
   for example `http://localhost:9080/EchoClientApplication/default.jsp`, and
   log in.

   Note: Log on using a user name and password that is in the WebSphere
   Application Server user registry that is used by the server where the client
   runs.
2. From the **Token type** list, select the token type to send from this client.
   Different parts of the Web services security management component are
   demonstrated depending on your selection.

Select token type	To demonstrate
No token	No credentials are sent.
Kerberos	WSSM generates a SAML 2.0 token internally from the JAAS subject. The STS exchanges it for a Kerberos token.

Select token type	To demonstrate
SAML 1.1	WSSM generates a SAML 1.1 token internally from the JAAS subject. No call is made to the STS.
SAML 2.0	WSSM generates a SAML 2.0 token internally from the JAAS subject. No call is made to the STS.
Tivoli Access Manager	WSSM generates a SAML 2.0 token internally from the JAAS subject. The STS exchanges it for a Tivoli Access Manager token.
Username	WSSM generates a SAML 2.0 token internally from the JAAS subject. The STS exchanges it for a Username token.
X.509	WebSphere Application Server generates an X.509 token. WSSM and the STS are not involved.

Selecting a token type automatically selects Echo Web service URL for that token type.

For example, if you select SAML 1.1, the Echo Web service URL is set to `http://localhost:9080/EchoApplication/services/EchoServiceSAML11`.

However, this URL can be modified. For example, if you are using TCPMonitor to capture traffic to and from the Echo client, you might set this URL to `http://localhost:51962/EchoApplication/services/EchoServiceSAML11.jsp`.

3. Click one of the function buttons to start the Web service.

Using the client functions

The Echo client presents several functions.

Using the "Echo" function

The **Echo** function configures the server application to return the text to the client unchanged.

Procedure

1. In the **Input** field, enter the text to be sent to the Echo server application.
2. Click **Echo** to configure the server to echo the text back to the client. The returned text shows in the **Result** field.

Using the "To Lower" function

The **To Lower** function configures the server application to convert the input text to lower case and then return it.

Procedure

1. In the **Input** field, enter the text to be sent to the Echo server application.
2. Click **To Lower** to configure the server to convert the input text to lower case and return the text back to the client. The returned text shows in the **Result** field.

Using the "To Upper" function

The **To Upper** function configures the server application to convert the input text to uppercase and then return it.

Procedure

1. In the **Input** field, enter the text to be sent to the Echo server application.
2. Click **To Upper** to configure the server to convert the input text to uppercase and return the text back to the client. The returned text shows in the **Result** field.

Using the "Who Am I" function

The **Who Am I** function instructs the server application to return the credentials of the client.

About this task

Click **Who Am I** to configure the server to return the credentials of the logged in user.

The **Who Am I** function returns the Java subject information that is available to the Echo Web service application. This information includes the principal names and the public and private credentials.

A WSSMCredential is included with the private credentials. This WSSMCredential contains the SAML 2.0 assertion that is returned by the trust service and is used to perform the JAAS login for the application. The assertion includes an authentication statement that contains the identity used to log in. The assertion also includes an attribute statement that contains the identity that was supplied by the Echo client.

Removing the Echo demonstration application

You can remove the Echo demonstration application if you do not need it anymore.

Procedure

1. Start the WebSphere Application Server administrative console and log on, if necessary.

   Note: Ensure that you are using the administrative console associated with the application server where the Web services security management component is installed.
2. Click **Applications** > **Enterprise Applications** to open the Enterprise Applications page.
3. Optional: If the application to be removed is running, select the application and then click **Stop**.
4. Select the application to be removed.
5. Click **Uninstall** to remove the application from WebSphere Application Server.
6. Click **OK** to confirm removing the application.
7. In the Messages pane at the top of the window, click **Save** to save your changes.
8. Restart the application server.

Appendix. Web services standards

Web services security for WebSphere Application Server is based on standards described in the Organization for the Advancement of Structured Information Standards (OASIS) Web services security (WS-Security) specification.

For additional information about Web services standards, see:

http://www.ibm.com/developerworks/webservices/standards/

For information regarding service-oriented architecture from IBM, see:

http://www.ibm.com/soa

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law :

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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 and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to

IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

If you are viewing this information in softcopy form, the photographs and color illustrations might not be displayed.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information; at www.ibm.com/legal/copytrade.shtml.

Adobe, Acrobat, PostScript and all Adobe-based trademarks are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Index

logging
   enabling 56
   wsdl2tam 43
login
   Echo client 22
   Echo service 23
login modules
   overview 16
   SAML assertion 16
   Tivoli Access Manager 17
   Username 17

M

module types
   supported 5
monitoring
   communications 55
   Echo application 88

N

notation
   environment variables xiv
   path names xiv
   typeface xiv

O

OASIS 93
object space
   creating 46
   Tivoli Access Manager overview 45
ordering publications xiii
overview
   component 1
   interactions with products 2
   Web service application 5

P

PassTicketSTSModule
   description 6
   keystore 50
path names, notation xiv
publications
   ordering xiii
   related xii

R

removing
   Echo demonstration application 91
running
   Echo demonstration application 59

S

SAML 1.1
   configuring WS-Security 66
SAML login module
   configuring for WebSphere
     Application Server 30
   description 16

SAML10STSModule
   description 6
   keystore 49
SAML11STSModule
   keystore 49
SAML20STSModule
   description 6
   keystore 49
SAMLA token consumer
   overview 15
   supported token types 15
SAMLTokenSTSModule
   description 6
scripts, modifying 39
shared library configuration 28
standards 93
STS module chains
   configuring for Echo 74, 80
   configuring for Echo client 83, 85
sts.call 9, 14
STSLTPATokenModule
   description 6
STSMapDefault
   description 6
STSMessageLogger
   description 6
STSTokenIVCred
   description 6

T

TAMAuthenticationSTSModule
   description 6
TAMAuthorizationSTSModule
   description 6
Tivoli Access Manager
   access control policy 48
   configuring WS-Security 68
   login module 17
   object space
     creating 46
     overview 45
   use with Echo demonstration
     application 61
Tivoli Access Manager login module
   configuring for WebSphere
     Application Server 31
Tivoli technical training xiii
token consumers
   overview 12
   SAMLA 15
   WSSM 13
token generator
   overview 7
   parameters 8
   supported token types 8
token module types 5
training, Tivoli technical xiii
trust service
   monitoring 55
   securing connection to WebSphere
     Application Server 34
typeface conventions xiv

U

uninstalling
   Echo demonstration application 91
unique.token.id.xpath 10, 15
Username
   configuring WS-Security 70
Username login module 17
   configuring for WebSphere
     Application Server 33
UsernameTokenSTSModule
   description 6
   keystore 49

V

variables in WebSphere Application
  Server 27
variables, notation for xiv

W

Web service application 51
   configuration overview 5
   preparing to use 4
Web services security management
  (WSSM)
   configuring for 51
Web services standards 93
WebSphere Application Server
   administrative console, starting 27
   callback handlers 11
   checking status 37
   class loader 29
   configuration 27
   configuring for cluster
     environment 29
   Java 2 security 33
   monitoring 55
   restarting 36
   SAML login module 30
   securing connection 34
   shared library 28
   Tivoli Access Manager login
     module 31
   Username login module 33
   variables 27
WS-Security
   configuring for Echo 64
wsdl2tam
   logging 43
   modifying 41
   sample 41, 42
   using 46
wsdl2tam command 47
WSSM (Web services security
  management)
   configuring for 51
WSSM token consumer
   overview 13
   parameters 14
   supported token types 13
WSSM token generator
   overview 7
wssm.properties
   modifying 39
   sample 40

X

IBM ®

Printed in USA