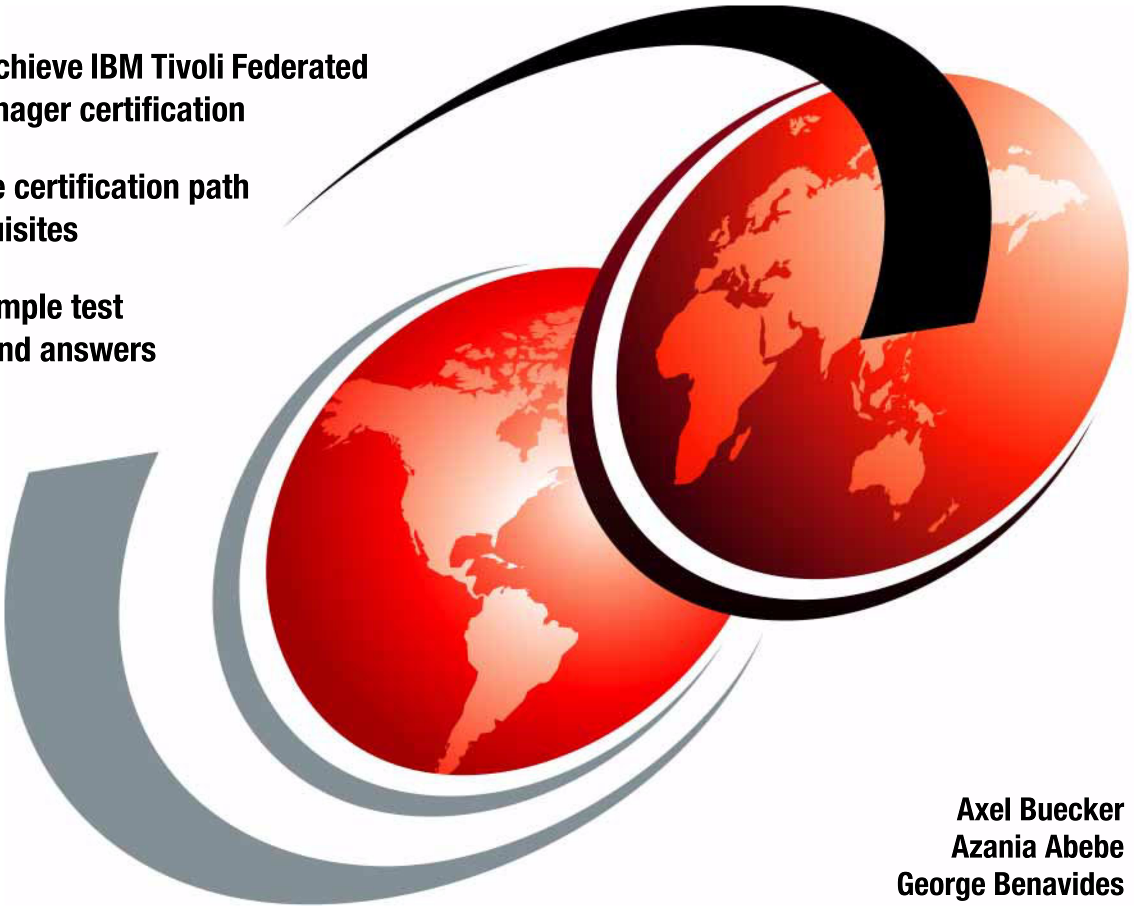
Tivoli. software

IBM

# Certification Study Guide Series: IBM Tivoli Federated Identity Manager 6.1



- Helps you achieve IBM Tivoli Federated Identity Manager certification
- Explains the certification path and prerequisites
- Includes sample test questions and answers

Axel Buecker
Azania Abebe
George Benavides

Redbooks

ibm.com/redbooks

International Technical Support Organization

# Certification Study Guide Series: IBM Tivoli Federated Identity Manager 6.1

September 2009

SG24-7783-00

**Note:** Before using this information and the product it supports, read the information in "Notices" on page vii.

**First Edition (September 2009)**

This edition applies to IBM Tivoli Federated Identity Manager 6.1.

# Contents

# Notices

# Preface

This IBM® Redbooks® publication is a study guide for the “IBM Certified Deployment Professional - IBM Tivoli® Federated Identity Manager V6.1” certification test, test number 000-891, and is meant for those who want to achieve IBM Certifications for this specific product.

The IBM Tivoli Federated Identity Manager Certification, offered through the Professional Certification Program from IBM, is designed to validate the skills required of technical professionals who work with the implementation of the IBM Tivoli Federated Identity Manager Version 6.1 product.

This book provides a combination of theory and practical experience needed for a general understanding of the subject matter. It also provides sample questions that will help in the evaluation of personal progress and provide familiarity with the types of questions that will be encountered in the exam.

This publication does not replace practical experience, and it is not designed to be a stand-alone guide for any subject. Instead, it is an effective tool that, when combined with education activities and experience, can be a very useful preparation guide for the exam.

## The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Austin Center.

**Axel Buecker** is a Certified Consulting Software IT Specialist at the ITSO, Austin Center. He writes extensively and teaches IBM classes worldwide on areas of software security architecture and network computing technologies. He holds a degree in Computer Science from the University of Bremen, Germany. He has 23 years of experience in a variety of areas related to workstation and systems management, network computing, and e-business solutions. Before joining the ITSO in March 2000, Axel worked for IBM in Germany as a Senior IT Specialist in Software Security Architecture.

**Azania Abebe** is a Certified Senior Security Consultant with the IBM Software Services, Tivoli Security, and Privacy Practice. He has extensive industry experience in the identity management space specializing in the delivery of Tivoli-based technologies and enterprise solutions around identity and access management. He has over 14 years combined experience in information technology and software development of enterprise applications. Currently, Azania focuses on architecting solutions and advising IT organizations on effective alignment of IT infrastructure, security requirements, and business objectives.

**George Benavides** works in the IBM Tivoli Access Manager and Federated Identity Manager support group in Austin, TX. He has over 15 years of industry experience. George graduated the University of Texas in Austin with M.S.E.E and B.S.E.E. degrees. He has also held various Developer positions in communications and security areas. Before joining the IBM support organization in Austin, George held a Developer position in AIX® security.

Thanks to the following people for their contributions to this project:

Emma Jacobs, Diane Sherman
International Technical Support Organization

Judy Green, Vladimir Jeremic, Neil Readshaw
IBM

## Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

**ibm.com**/redbooks/residencies.html

## Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  **ibm.com**/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

# Certification overview

In this chapter we provide an overview of the skill requirements needed to obtain an IBM Advanced Technical Expert certification. The following sections are designed to provide a comprehensive review of specific topics that are essential for obtaining the certification:

- IBM Professional Certification Program
- Tivoli Federated Identity Manager 6.1 certification
- Recommended educational resources

## 1.1 IBM Professional Certification Program

Having the right skills for the job is critical in the growing global marketplace. IBM Professional Certification, designed to validate skill and proficiency in the latest IBM solution and product technology, can help provide that competitive edge. The IBM Professional Certification Program Web site is available at:

http://www.ibm.com/certify/index.shtml

The IBM Professional Certification Program offers a business solution for skilled technical professionals seeking to demonstrate their expertise to the world.

The program is designed to validate your skills and demonstrate your proficiency in the latest IBM technology and solutions. In addition, professional certification can help you excel at your job by giving you and your employer confidence that your skills have been tested. You may be able to deliver higher levels of service and technical expertise than non-certified employees and move on a faster career track. Professional certification puts your career in your control.

The certification requirements are difficult, but not impossible. Certification is a rigorous process that differentiates you from everyone else.

The mission of IBM Professional Certification is to:

- Provide a reliable, valid, and fair method of assessing skills and knowledge.
- Provide IBM with a method of building and validating the skills of individuals and organizations.
- Develop a loyal community of highly skilled certified professionals who recommend, sell, service, support, and use IBM products and solutions.

The IBM Professional Certification Program has developed certification role names to guide you in your professional development. The certification role names include IBM Certified Specialist, IBM Certified Solutions/Systems Expert, and IBM Certified Advanced Technical Expert for technical professionals who sell, service, and support IBM solutions.

For technical professionals in application development, the certification roles include IBM Certified Developer Associate and IBM Certified Developer. IBM Certified Instructor certifies the professional instructor.

The IBM Professional Certification Program provides you with a structured program leading to an internationally recognized qualification. The program is designed for flexibility by enabling you to select your role, prepare for and take tests at your own pace, and, in some cases, select from a choice of elective tests best suited to your abilities and needs. Some roles also offer a shortcut by giving credit for a certification obtained in other industry certification programs.

You might be a network administrator, systems integrator, network integrator, solution architect, solution developer, value-added reseller, technical coordinator, sales representative, or educational trainer. Regardless of your role, you can start charting your course through the Professional Certification Program from IBM today.

## 1.1.1 Benefits of certification

Certification is a tool to help objectively measure the performance of a professional on a given job at a defined skill level. Therefore, it is beneficial for individuals who want to validate their own skills and performance levels, their employees, or both. For optimum benefit, the certification tests must reflect the critical tasks required for a job, the skill levels of each task, and the frequency by which a task has to be performed. IBM prides itself in designing comprehensive, documented processes that ensure that IBM certification tests remain relevant to the work environment of potential certification candidates.

In addition to assessing job skills and performance levels, professional certification may also provide such benefits as:

- For employees:
  - Promotes recognition as an IBM Certified Professional
  - Helps to create advantages in interviews
  - Assists in salary increases, corporate advancement, or both
  - Increases self-esteem
  - Provides continuing professional benefits
- For employers:
  - Measures the effectiveness of training
  - Reduces course redundancy and unnecessary expenses
  - Provides objective benchmarks for validating skills
  - Makes long-range planning easier
  - Helps to manage professional development
  - Aids as a hiring tool
  - Contributes to competitive advantage
  - Increases productivity, morale, and loyalty
- For Business Partners and consultants:
  - Provides independent validation of technical skills
  - Creates competitive advantage and business opportunities
  - Enhances prestige of the team
  - Contributes to IBM requirements for various IBM Business Partner programs

Specific benefits might vary by country (region) and role. In general, after you become certified, you should receive the following benefits:

- Industry recognition

  Certification can accelerate your career potential by validating your professional competency and increasing your ability to provide solid, capable technical support.

- Program credentials

  As a certified professional, you receive (through e-mail) your certificate of completion and the certification mark associated with your role for use in advertisements and business literature. You may also request a hardcopy certificate, which includes a wallet-size certificate.

  The Professional Certification Program from IBM acknowledges the individual as a technical professional. The certification mark is for the exclusive use of the certified individual.

- Ongoing technical vitality

  IBM Certified Professionals are included in mailings from the IBM Professional Certification Program.

### 1.1.2 Tivoli Software Professional Certification

The IBM Tivoli Professional Certification Program offers certification testing that sets the standard for qualified product consultants, administrators, architects, and partners.

The program also offers an internationally recognized qualification for technical professionals who are seeking to apply their expertise in today's complex business environment. The program is designed for those who implement, buy, sell, service, and support Tivoli solutions and who want to deliver higher levels of service and technical expertise.

Whether you are a Tivoli customer, partner, or technical professional wanting to put your career on the fast track, you can start your journey to becoming a Tivoli Certified Professional today.

#### Benefits of being Tivoli certified

Tivoli Certification has the following benefits:

- For the individual:
  - IBM Certified certificate and use of logos on business cards
  - Recognition of your technical skills by your peers and management
  - Enhanced career opportunities
  - Focus for your professional development

- For the Business Partner:
  - Confidence in the skills of your employees
  - Enhanced partnership benefits from the Business Partner Program
  - Higher rates for billing out your employees
  - Stronger customer proposals
  - Demonstration of the depth of technical skills available to prospective customers
- For the customer:
  - Confidence in the services professionals handling your implementation
  - Ease of hiring competent employees to manage your Tivoli environment
  - Ease of selecting a Tivoli Business Partner that meets your specific needs
  - Enhanced return on investment (ROI) through more thorough integration with Tivoli and third-party products

## Certification checklist

To pursue certification, follow the steps in this checklist:

1. Select the certification you would like to pursue.
2. Determine which tests are required by reading the certification role description.
3. Prepare for the test by using the following resources:
   - Test objectives
   - Recommended educational resources
   - Sample assessment test
   - Other reference materials
   - Opportunities for experience

   **Note:** These resources are available from each certification description page and from the test information page.
4. Register to take a test, by contacting one of our worldwide testing vendors:
   - Prometric
   - Pearson Virtual University Enterprises (VUE)

   **Note:** When providing your name and address to the testing vendor, be sure to specify your name exactly as you would like it to appear on your certificate.
5. Take the test. Be sure to keep the Examination Score Report provided upon test completion as your record of taking the test.

**Note:** After you take the test, the results and demographic data (such as name, address, e-mail, and phone number) are sent from the testing vendor to IBM for processing (allow two to three days for transmittal and processing). After all the tests required for a certification are passed and received by IBM, your certificate will be issued.

6. Repeat steps 3 - 5 until all required tests are successfully completed for the certification. If you must meet additional requirements (such as another vendor certification or exam), follow the instructions on the certification description page to submit these requirements to IBM.
7. After you meet the requirements, you will be sent an e-mail asking you to accept the terms of the IBM Certification Agreement.
8. Upon your acceptance, you receive an e-mail with the following deliverables:
   - A Certification Certificate in PDF format, which can be printed in either color or black and white
   - A set of graphic files containing the IBM Professional Certification mark associated with the certification achieved
   - Guidelines for the use of the IBM Professional Certification mark
9. To avoid an unnecessary delay in receiving your certificate, ensure that your current e-mail is on file by keeping your profile up to date. If you do not have an e-mail address on file, your certificate will be sent by postal mail.

Certificates are sent by e-mail. However, you may also contact IBM at the following e-mail address to request a paper copy of the certificate, including a laminated wallet-sized card:

mailto:certify@us.ibm.com

**Note:** IBM reserves the right to change or delete any portion of the program, including the terms and conditions of the IBM Certification Agreement, at any time without notice. Some certification roles offered through the IBM Professional Certification Program require recertification.

## 1.2 Tivoli Federated Identity Manager 6.1 certification

In this section, we categorize the certification process for IBM Tivoli Federated Identity Manager.

**Important:** IBM offers the following promotion code, which is good for a 15% discount on the indicated Tivoli certification exams if taken at any Prometric testing center:

- Code: 15T891
- Percentage off: 15%
- Valid for exams: 000-891

### 1.2.1 Job description and target audience

An IBM Certified Deployment Professional - Tivoli Federated Identity Manager V6.1 is a technical professional who is responsible for the planning, designing, customizing, testing, troubleshooting, and documenting of solutions for IBM Tivoli Federated Identity Manager V6.1. This individual is expected to perform these tasks with limited assistance from peers, product documentation, and support resources.

### 1.2.2 Required prerequisites

The required prerequisites necessary for passing Certification Test 000-891 include the following abilities:

- Assess customer's architecture and solution design documentation.
- Analyze the deployment environments.
- Assist in project plan development.
- Apply federation management concepts (federated identity management, Web services security management, federated provisioning).
- Perform basic installations of the prerequisite applications (IBM Tivoli Directory Integrator, LDAP/DB2®, WebSphere® Application Server, Tivoli Access Manager for e-business).
- Describe the Tivoli Federated Identity Manager features and components.
- Configure product and component integration points such as Tivoli Directory Integrator, WebSphere Application Server, Common Auditing and Reporting Service (CARS), Tivoli Access Manager for e-business.
- Install and configure federated single sign-on (F-SSO), Web services security management, and federated provisioning services.

- Identify key components of F-SSO protocols.
- Demonstrate working knowledge of the Tivoli Federated Identity Manager subsystems.
- Deploy Tivoli Federated Identity Manager customer solutions leveraging combinations of the F-SSO, Web services security management, and federated provisioning services.
- Troubleshoot Tivoli Federated Identity Manager services.
- Understand basic Web page development fundamentals (including security issues).
- Have a working knowledge of:
  - Operating systems
  - Server hardware and networking technologies
  - System administration of UNIX®, Windows®, or Linux® operating systems
  - XML terms and concepts including XSLT, XML DSig, and XML encryption
  - SOAP terms and concepts including WS Security, WS Trust, WSDL, Web service deployment
  - LDAP (IBM Tivoli Directory Server)
  - Basic editors such as vi
  - WebSphere Application Server (administration console, clustering)
  - IBM Tivoli Directory Integrator and JavaScript™
  - IBM Tivoli Access Manager for e-business
  - F-SSO protocols (SAML, WS Federation, Liberty)
  - Web service application deployment
  - Federated provisioning
  - Common Auditing and Reporting Services (CARS) subsystem shipped with Tivoli Federated Identity Manager
  - Third-party XML firewall/gateways
- Have programming and scripting experience including JSP™, ActiveX®, Java™.
- Have experience and knowledge with TCP/IP networking principles including SSL.
- Have general knowledge of security concepts including key management and PKI (Public Key Infrastructure) fundamentals.

- Work with environment variables including: local variables, exported variables, HOME, PATH, subshells.
- Understand security policy management concepts.
- Have familiarity with:
  - Additional layers of the protocol stack and associated wireless protocols: WTLS, WML, WAP
  - Web service application tooling such as IBM Rational® Application Developer, IBM WebSphere Studio Application Developer
  - Scripting of wsadmin and jython
  - WebSphere Application Server Dynacache
  - TCPMon
  - Data Center terms, concepts and methodologies: load balancing, firewalls, switching and routing (Level2, Level3)
  - Common Event Infrastructure

## 1.2.3 Test 000-891 objectives

Let us look more closely at the seven objective areas for this test:

- Section 1: Plan for federation
- Section 2: Plan for federated single sign-on
- Section 3: Plan for Web services security management
- Section 4: Install infrastructure and components
- Section 5: Configure
- Section 6: Test
- Section 7: Troubleshoot

### Section 1: Plan for federation

The section provides information about the planning area of the test:

- Given a set of architecture documents, review the scenario described, review the customer's use cases, identify the Tivoli Federated Identity Manager functions, and identify the role of customer in the federation so that a valid use case and scenario document is prepared, which details the Tivoli Federated Identity Manager functions and protocols in relation to the customer's role in the federation. The emphasis is on being able to perform the following steps:
  a. Review scenario described.
  b. Review use cases.
  c. Identify Tivoli Federated Identity Manager function.
  d. Identify customer role (identity provider or service provider).

- Given a valid use case and scenario document that describes the customer's roles and customer's usage requirements (for example, performance requirements), identify how the Tivoli Federated Identity Manager components map to the customer's environment so that the details of the customer environment are qualified and required platforms are listed. The emphasis is on being able to perform the following steps:

  a. Identify authentication service (HTTP, direct).
  b. Identify session management (HTTP).
  c. Identify authorization services.
  d. Identify alias service.
  e. Identify F-SSO identity services.
  f. Identify identity management solution providing endpoints.
  g. Determine platforms.
  h. Identify *point of contact* (SOAP) for mobile, WAP gateway, LECP/ECP.

- Given the output of the mapping of the customer requirements to Tivoli Federated Identity Manager services and a list of the required platforms, determine the number of machines (and if any additional IT infrastructure is needed) so that a list of target machines is produced. The emphasis is on being able to perform the following steps:

  a. Get permission to install.
  b. Determine machine numbers and specifications.
  c. Reconcile, and determine additional platforms and IT resources.

- Given the customer's security policy, determine the audit and report methodology (using either CARS or audit log), F-SSO, Web services provisioning, and Web services security management security policies so that audit log configuration is defined and high security level policy is outlined detailing signed components, encryption, authorization, authentication, and transport security for each Tivoli Federated Identity Manager function. The emphasis is on being able to perform the following steps:

  a. Determine audit log policy.
  b. Determine F-SSO security requirements.
  c. Determine WS Provisioning security requirements.
  d. Determine Web services security management security policy.

- Given the customer's use cases, selected federation business partner identities, and target number of federation business partners, determine federation business partner functionality, evaluate federation business partner's requirements, and define test environment so that a matrix of federation business partner by functionality and requirements is created and generate a test plan. The emphasis is on being able to perform the following steps:

  a. Determine federation business partner functionality.
  b. Evaluate federation business partner's security policy.

   c. Determine federation business partner ID map requirements.
   d. For Web services security management, determine WS trust names pace.
   e. Define customer-partner test environment.
   f. Build test drivers.

- Given a matrix of federation business partner by functionality and requirements, list of target machines, and details of customer environment, map Tivoli Federated Identity Manager functions to Tivoli Federated Identity Manager components and target machines so that an installation plan is created. The emphasis is on being able to identify Tivoli Federated Identity Manager functions, components, and target match.
- Given a list of federation business partners with security policy and a matrix of federation business partner by functionality, define the federations so that each federation business partner is assigned to a federation and the function of each federation is listed. The emphasis is on being able to perform the following steps:

   a. Map federation business partners to federations.
   b. Create new federations if required.

## Section 2: Plan for federated single sign-on

The section provides information about the planning for federated single sign-on (F-SSO) area of the test:

- Given a mapping of F-SSO partners to federations, a definition of each federation, the F-SSO customer-partner security policy, and the additional attributes require in the F-SSO tokens, refine the F-SSO details so that the parameters for the customer's self-configuration and high level mapping of attribute requirements are documented for each F-SSO federation. The emphasis is on being able to perform the following steps:

   a. Define or determine encryption and signing requirements for messages.
   b. Determine encryption requirements for messages.
   c. If required, determine token types.
   d. Determine token security parameters.
   e. Determine *message parameters*: lifetime, nonce, and so on.
   f. Define protocol and federation-specific endpoints.
   g. Determine ID mapping rules (high level).

## Section 3: Plan for Web services security management

The section provides information about the planning for Web services security management area of the test:

- Given a description of the Web services environment and applications, define the Web services point of contact, the type of service, and identify the login method for each application so that a list of applications to be deployed in

Web services security management is generated. The emphasis is on being able to perform the following steps:

a. Identify the Web services point of contact (for example, XML framework, WSGW, and so on).

b. Identify the type of Web service (for example, SOAP/HTTP, SCAP/JMS, RMI/IIOP, and so on).

c. Identify whether the Web service is considered an endpoint or an intermediary.

d. If the Web service is considered an endpoint, is a login required?

e. If the Web service is considered an intermediary, is a token exchange required?

f. Determine the list of applications to be deployed with Web services security management.

- Given a list of Web services security management (WSSM) partners, the customer-partner WSSM security policy, and the information required to be in the incoming token (included with partner's Web services request), determine the requirements for authentication and authorization for each application and for each federation business partner and identify the applications the federation business partner can access so that the parameters of the local configuration of the WSSM federation, application side, and partner side of WSSM, and high level mapping of the requirements and rules are defined. The emphasis is on being able to perform the following steps:

  a. If required, determine the application token type versus login.

  b. Determine requirements for encrypting messages by application.

  c. Define or determine requirements for signing messages by application.

  d. If required, determine requirements for encrypting or signing *output* tokens.

  e. Determine authorization required by application.

  f. Define applications available to federation business partners.

  g. Define ID mapping rules (high level) by federation business partner.

  h. Determine requirements for encryption input tokens by federation business partner.

  i. Determine requirements for signing input tokens by federation business partner.

  j. If required, determine federation business partner output token type.

## Section 4: Install infrastructure and components

The section provides information about the installation area of the test:

- Given the WebSphere Application Server deployment strategy, install media, cluster information, and architecture document, run the WebSphere Application Server installation, crate the application server profile, create the deployment manager profile, a WebSphere Application Server cluster, a replication domain, and add the application server to the cluster so that WebSphere Application Server is installed and configured for Tivoli Federated Identity Manager. The emphasis is on being able to perform the following steps:

  a. Install WebSphere Application Server.
  b. Create an application server profile.
  c. If using clustering, create a deployment manager profile.
  d. Create a profile.
  e. If clustering, create a cluster.
  f. If clustering, add other servers to the cluster.

- Given the architecture document, directory information, Tivoli Access Manager for e-business installation, SSL keys, and proper access, install patches, GSKit, Access Manager Runtime Environment (AMRTE) file sets, and run the **pdconfig** command with the correct information so that WebSEAL is successfully installed and configured into the Tivoli Access Manager for e-business domain. The emphasis is on being able to perform the following steps:

  a. Identify OS patches to install.
  b. Install OS patches.
  c. Install GSKit.
  d. Install AMRTE.
  e. Install file sets.
  f. Configure WebSEAL into Tivoli Access Manager for e-business domain.

- Given the Integrated Solutions Console (ISC) installation media, verify that the LDAP server is running and run the ISC installation so that the ISC is property installed and configured. The emphasis is on being able to perform the following steps:

  a. Verify that the LDAP server is running.
  b. Install Tivoli Federated Identity Manager ISC.

- Given the Tivoli Federated Identity Manager media, ISC is installed and configured, and WebSphere Application Server is running, run the install program for the Tivoli Federated Identity Manager Runtime so that the Tivoli Federated Identity Manager Runtime is successfully installed.

The emphasis is on being able to perform the following steps:

a. Verify that LDAP is running.
b. Install Federated Identity Manager Runtime.
c. Create domain.
d. Deploy Tivoli Federated Identity Manager Runtime.

- Given the installation media, install the file sets to successfully perform an IBM Tivoli Directory Integrator installation. The emphasis is on being able to install Tivoli Directory Integrator file sets.
- Given the architecture document, the WebSphere Application Server Network Deployment install media, and the required patches, install WebSphere Application Server Network Deployment and apply the required patches to create a new WebSphere Application Server application profile and install the server integration business Web services components to create a configured Web Services Gateway. The emphasis is on being able to perform the following steps:

  a. Install WebSphere Application Server Network Deployment.
  b. Create a new application profile.
  c. Install patches.
  d. Install the Service Integration Business Web Services components.

- Given the need for Common Auditing and Reporting Services (CARS) and the installation media, confirm all prerequisites have been met, run CARS install, so that CARS is installed. The emphasis is on being able to perform the following steps:

  a. Install IBM DB2.
  b. Configure the DB2 instance.
  c. Install and Configure CARS Server.
  d. Configure the Common Event Infrastructure in WebSphere Application Server.
  e. Install the CARS client.
  f. Configure Tivoli Access Manager for e-business for CARS.
  g. Verify event data within DB2.
  h. Install and Configure Crystal Reports (including the prebuilt Tivoli Access Manager for e-business reports).
  i. Generate Tivoli Access Manager for e-business reports with Crystal Reports.

## Section 5: Configure

The section provides information about the configuration area of the test:

- Given the LDAP access information and the name of the new alias service and suffix, add the new suffix and restart WebSphere Application Server to have LDAP configured for Tivoli Federated Identity Manager. The emphasis is on being able to perform the following steps:

  a. Stop the LDAP service.
  b. Add LDAP suffix for alias service.
  c. Restart the LDAP service.

- Given the attribute requirements for applications, roles, user of group definitions, attribute schema, and XSLT authoring tools, use the XSLT tool to successfully write and run a mapping rule. The emphasis is on being able to perform the following steps:

  a. Write an XSLT (mapping) rule.
  b. Run XSLT (mapping) tool.

- Given the WebSEAL information, company information, protocol, role, token requirement, protocol specific configuration, and defined mapping rules, successfully create and configure a federation. The emphasis is on being able to perform the following steps:

  a. Log in to the Integrated Solutions Console (ISC) and create a federation.
  b. Follow the Federation Creation wizard and enter the appropriate data.
  c. Send meta data to federation business partner.

- Given the federation business partner's meta data and specific configuration, log in to console, define a federation business partner and enable a federation business partner for a configured working partner. The emphasis is on being able to perform the following steps:

  a. Log in to the ISC and add a federation business partner.
  b. Follow the Add Partner wizard.
  c. Enable federation business partner.

- Given a federation business partner's client certificate configuration, certificate authority certification for HTTPS connection, security requirements for WebSEAL to WebSphere Application Server communication, WebSphere Application Server port information, role, federation name, Tivoli Federated Identity Manager F-SSO endpoint, and user attribute information, configure WebSEAL for Tivoli Federated Identity Manager so that a working WebSEAL configuration for a specific federation is created. The emphasis is on being able to perform the following steps:

  a. Configure tag value.

  b. Use the TFIMCFG tool to create a junction, configure EAI, and assign ACLs.

c. If the role is *service provider*, modify the `login.html` page to point to single sign-on endpoint.

d. Configure single logout endpoint.

e. Import federation business partner client certificates into WebSEAL keystore.

f. Increase WebSEAL POST cache size.

g. Implement a basic authentication user provisioning; create users as Tivoli Access Manager for e-business users at identity provider side.

- Given the architecture document, Tivoli Federated Identity Manager Application Developer Kit (ADK) and Java Development Tools, write, test and install the code, so that custom code is successfully created to meet the customer's requirements. The emphasis is on being able to perform the following steps:

  a. Write code.
  b. Test code.
  c. Install code.

- Given the architecture requirements, write, test and install custom token module, so that support is provided for a custom token type. The emphasis is on being able to perform the following steps:

  a. Write custom token code.
  b. Test custom token code.
  c. Install custom code.

- Given the required token types, attributes required, federation business partner keys, self keys, and configured mapping rules, log in to the ISC and add a WSSM partner. Follow the wizard and enter the required data, so that a configured WSSM partner is created. The emphasis is on being able to perform the following steps:

  a. Log in to the ISC and add a WSSM partner.
  b. Follow the Add WSSM Partner wizard and enter the required data.

- Given the trust service endpoint information, the application WSDL, the required application token types, the customer application, required WebSphere Application Server patches, and the WSDL2TFIM and WSDL2TAM tools, configure Tivoli Federated Identity Manager WSSM in WebSphere Application Server to create a deployed application secured by WSSM. The emphasis is on being able to perform the following steps:

  a. Configure a JAAS login module for Security Assertion Markup Language (SAML).

  b. Create WebSphere Application Server shared library for WSSM classes.

  c. Configure WSSM PDJRTE.

   d. Deploy customer application.

   e. Run WSDL2TFIM and WSDL2TAM tools.

   f. Configure Tivoli Access Manager for e-business policy.

   g. Apply WebSphere Application Server patches.

- Given the architecture document, registry information, and Tivoli Directory Integrator toolkit, write, test, and install code for a successful development of custom code. The emphasis is on being able to perform the following steps:

   a. Write federated provisioning code.
   b. Test federated provisioning code.
   c. Install federated provisioning code.

- Given the Tivoli Access Manager for e-business and WebSphere Application Server environment information and editor, update the Tivoli Directory Integrator AssemblyLine properties and the provisioning service endpoint to successfully update the provisioning configuration. The emphasis is on being able to perform the following steps:

   a. Update provisioning service endpoint custom property.
   b. Update AssemblyLine properties and constraints.

- Given Tivoli Federated Identity Manager and CARS are installed, configure Tivoli Federated Identity Manager to send audit events to the CARS server, so that CARS can be used by Tivoli Federated Identity Manager. The emphasis is on being able to perform the following steps:

   a. Import the CARS server root signer certificate to the Tivoli Federated Identity Manager keystore.

   b. In ISC, navigate to domain management and display the audit settings.

   c. Enable the audit function for using CARS.

   d. Set up the SSL keystore.

   e. Select the type of authentication: basic auth or no auth required.

## Section 6: Test

The section provides information about the solution testing area of the test:

- Given a configured Tivoli Federated Identity Manager environment with federated single sign-on (F-SSO), authenticate with the identity provider, and connect to the linked account at the service provider, so that there is a working Tivoli Federated Identity Manager environment with F-SSO. The emphasis is on being able to perform the following steps

   a. Authenticate with the identity provider.
   b. Connect to a linked account at the service provider.
   c. Test and verify single sign-on and account federation (Liberty, SAML 2.0).

d. Test and verify single sign-on (push, pull).
e. Test and verify HTTP-redirect, SOAP-HTTP profiles.
f. Test and verify Liberty RNI/FT and Name NIM profiles.
g. Test and verify *where are you from?*
h. Test and verify single logout (local, global).

- Given a WSSM installed and configured environment and a deployed Web services application with WS Security turned on, run the Web services application, and evaluate the results to successfully test the Web service application with WSSM enabled. The emphasis is on being able to perform the following steps:

  a. Run Web service application.
  b. Test unauthorized user.
  c. Test invalid password.
  d. Test encrypted Web services invocation.
  e. Test signed Web services invocation.
  f. Test signed and encrypted Web services invocation.
  g. Test invocation of federation business partner side of trust chain.
  h. Test that input token is valid (format, encrypt, signing).
  i. Test mapping rules.
  j. Test authorization decision regarding required input in IV-CRED.

- Given that Tivoli Federated Identity Manager is configured with WS Provisioning, the Tivoli Directory Integrator AssemblyLines running at both identity provider and service provider, create a local provisioning trigger at an identity provider, so that a local identity is provisioned at the service provider. The emphasis is on being able to perform the following steps:

  a. Provisioning request to create a test user.
  b. Provisioning request to modify attributes for a test user.
  c. Provisioning request to remove a test user.

## Section 7: Troubleshoot

The section provides information about the solution troubleshooting of the test:

- Given that IBM Tivoli Directory Server is installed, perform a test LDAP search, check for errors in `ibmslapd.log`, check for configuration in `ibmslapd.conf`, verify that the LDAP service is listening on the proper SSL/non-SSL parts, and check for proper ACLs so that IBM Tivoli Directory Server is integrated with Tivoli Federated Identity Manager and working properly. The emphasis is on being able to perform the following steps:

  a. Perform a test LDAP search.

  b. Check for errors in the `iblslapd.log file.`

  c. Check the `ibmslapd.conf` file for valid configuration.

   d. Verify that an LDAP service is listening on the proper ports for SSL and non-SSL communication.

   e. Check for proper ACLs.

- Given that WebSphere Application Server is installed, check for errors in the WebSphere logs, check for memory used by the Java process of WebSphere, check for the status of deployed applications, and check security settings validity for WebSphere Application Server and deployed application so that WebSphere Application Server is properly integrated with Tivoli Federated Identity Manager and is working. The emphasis is on being able to perform the following steps:

   a. Check for WebSphere Application Server logs.

   b. Check for memory used by Java processes of WebSphere Application Server.

   c. Check for status of deployed applications.

   d. Check the WebSphere Application Server and deployed application security configuration.

   e. Debug clustering issues (*dynacache*).

- Given that IBM Tivoli Access Manager for e-business is installed and configured, verify that WebSEAL is communicating with the Policy Server, collect debug information using Tivoli Access Manager for e-business trace facility, and isolate and qualify the problem so that Tivoli Access Manager for e-business is integrated with Tivoli Federated Identity Manager and is working. The emphasis is on being able to perform the following steps:

   a. Verify that WebSEAL is communicating with the policy server.

   b. Collect and debug information using Tivoli Access Manager for e-business trace facilities.

   c. Isolate problem.

   d. Qualify the problem.

- Given that the Integrated Service Console (ISC) is installed and configured, verify that the ISC login is available and verify connection with LDAP, so that the ISC is integrated with Tivoli Federated Identity Manager and is working. The emphasis is on being able to perform the following steps:

   a. Verify ISC login page is available.
   b. Verify connection with LDAP.
   c. Look in the ISC logs.
   d. Verify that LDAP is correctly configured.

- Given that the Tivoli Federated Identity Manager *Trust Service* is installed and configured, turn on tracing and review the trace logs for errors and stack traces, so that the Tivoli Federated Identity Manager Trust Service is working. The emphasis is on being able to perform the following steps:

a. Turn on tracing.

b. Review the trace logs for errors and stack traces.

c. Check the WebSphere Application Server/Web Services Gateway endpoint receiving the SOAP request.

- Given that Tivoli Federated Identity Manager is configured for F-SSO, turn on Tivoli Federated Identity Manager tracing for the configured F-SSO protocol and review the tracing data for errors and stack traces, so that Tivoli Federated Identity Manager F-SSO is working. The emphasis is on being able to perform the following steps:

  a. Turn on Tivoli Federated Identity Manager tracing for configured F-SSO protocol.

  b. Review tracing data and stack traces.

- Given Tivoli Federated Identity Manager is configured for Web services security management (WSSM), run the **tcpmon** command and check the output, check time stamps on tokens, and verify signatures, so that the Tivoli Federated Identity Manager configuration of WSSM is working. The emphasis is on being able to perform the following steps:

  a. Run **tcpmon** and check output.
  b. Check time stamps on tokens.
  c. Verify signatures (if enabled).

- Given that *federated provisioning* and Tivoli Directory Integrator are installed and configured, isolate and analyze the message generated from Tivoli Federated Identity Manager for federated provisioning, check the communication between Tivoli Federated Identity Manager and the Tivoli Directory Integrator server, check that the WS Provisioning connector is in server mode and is enabled and running, and check the SOAP connector is configured correctly so that federated provisioning is working. The emphasis is on being able to perform the following steps:

  a. Isolate and analyze the message generated from Tivoli Federated Identity Manager for federated provisioning.

  b. Check the communication between Tivoli Federated Identity Manager and the Tivoli Directory Integrator server.

  c. Check that the WS Provisioning connector is in server mode and is enabled and running.

  d. Check the DSMLv2 connector is in add/update mode and is enabled.

- Given that the Tivoli Federated Identity Manager configuration has been modified and saved, back up the Tivoli Federated Identity Manager configuration so that the Tivoli Federated Identity Manager configuration is successfully restored. The emphasis is on being able to back up Tivoli Federated Identity Manager configuration.
- Given that Tivoli Federated Identity Manager installation has been customized, document only customizations, so that there are documented Tivoli Federated Identity Manager customizations. The emphasis is on being able to document the customization.

## 1.3 Recommended educational resources

Courses and publications are offered to help you prepare for certification tests. Although the courses are recommended before taking a certification test, they are not required.

### 1.3.1 Courses

This section provides information about the currently available or planned Tivoli Federated Identity Manager courses. Refer to the Tivoli software education Web site to find the appropriate courses and education delivery vendor for each geography, available at:

`http://www.ibm.com/software/tivoli/education`

If you want to purchase Web-based training courses or are unable to locate a Web-based course or classroom course at the time and location you desire, contact one of our delivery management teams:

- Americas:

  `mailto:tivamedu@us.ibm.com`
- EMEA:

  `mailto:tived@uk.ibm.com`
- Asia-Pacific:

  `mailto:tivtrainingap@au1.ibm.com`

**Note:** Course offerings are continuously being added and updated. If you do not see courses listed in your geographical location, contact the delivery management team.

General training information is available at the following Web site:

`http://ibm.com/training`

Also, refer to the test preparation roadmap for Tivoli Federated Identity Manager:

`http://www.ibm.com/certify/tests/map891.shtml`

## IBM Tivoli Federated Identity Manager 6.1 Deployment and Administration

IBM Tivoli Federated Identity Manager 6.1 employs a loosely-coupled model for managing identities and access across security, company, or organizational domains. Instead of replicating identity and security administration in multiple places, IBM Tivoli Federated Identity Manager provides a simple trust model for managing identities and access to information and services. IBM Tivoli Federated Identity Manager also provides policy-based integrated security management for federated Web services. In this 50% hands-on course, students learn to deploy and administer secure business-to-business and single sign-on federated environments.

### *Course duration*

This is a five-day, classroom course.

### *Objectives*

After completing this course, you should be able to:

- Plan the appropriate federated architecture based on partner goals, environment, and user attributes.
- Install and configure IBM Tivoli Federated Identity Manager 6.1.
- Describe how Key Management is implemented in IBM Tivoli Federated Identity Manager 6.1.
- Configure a SAML 1.1 single sign-on federation.
- Configure a WS-Federation single sign-on federation.
- Configure the alias service.
- Configure a SAML 2.0 single sign-on federation.
- Describe how Trust Service Modules are configured.
- Describe the different Trust Service Plug-ins included in IBM Tivoli Federated Identity Manager 6.1.
- Configure Web Services Security Manager for business-to-business transactions.
- Troubleshoot IBM Tivoli Federated Identity Manager issues.
- Plan for installation of Tivoli Federated Identity Manager 6.1 upgrade patches.
- Describe Tivoli Federated Identity Manager's role in an SOA.

### *Outline*

The course outline is as follows:

1. Introduction to IBM Tivoli Federated Identity Manager
2. IBM Tivoli Federated Identity Manager Prerequisites
3. IBM Tivoli Federated Identity Manager Architecture
4. Installing IBM Tivoli Federated Identity Manager and Integrated Solution Console
5. Configuring a SAML single sign-on federation
6. Configuring a WS-Federation single sign-on federation
7. Configuring the Alias Service
8. Configuring a SAML 2.0 single sign-on federation
9. Web Services Security Manager Introduction and Installation
10. Web Services Security Manager Configuration
11. Troubleshooting IBM Tivoli Federated Identity Manager

### *Required skills*

Before taking this course, you should possess knowledge and skills in:

- Basic Linux administrative skills
- IBM WebSphere Application Server 6.1
- IBM Tivoli Access Manager for e-business 6.0
- HTTP(S)
- Web servers
- Web services
- LDAP
- XML
- SAML 1.1
- SAML 2.0
- WS-Federation
- SOAP

## 1.3.2 Publications

IBM Tivoli Federated Identity Manager guides and Redbooks publications are useful tools for preparing to take Test 000-891.

## Product documentation

Refer to the following guides as a source of information:

- IBM Tivoli Federated Identity Manager information center, (only available online as an HTML version, either on the Tivoli publications Web site or through your local installation)

  `http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tivoli.fim.doc/welcome.htm`

- *IBM Tivoli Federated Identity Manager Installation Guide Version 6.1.1*, GC32-1667
- *IBM Tivoli Federated Identity Manager Configuration Guide Version 6.1.1*, GC32-1668
- *IBM Tivoli Federated Identity Manager Auditing Guide 6.1.1*, GC32-2287
- *IBM Tivoli Federated Identity Manager Problem Determination Guide Version 6.1.1*, GC32-2288
- *IBM Tivoli Federated Identity Manager Single Sign-on Guide Version 6.1.1*, GC32-0168
- *IBM Tivoli Federated Identity Manager Web Services Security Management Guide Version 6.1.1*, GC32-0169

## IBM Redbooks

Refer to the following publications related to Tivoli Federated Identity Manager:

- *Enterprise Security Architecture Using IBM Tivoli Security Solutions*, SG24-6014

  This publication reviews the overall Tivoli Enterprise Security Architecture. It focuses on the integration of audit and compliance, access control, identity management, and federation throughout extensive e-business enterprise implementations. The available security product diversity in the marketplace challenges everyone in charge of designing single secure solutions or an overall enterprise security architecture. With Access Manager, Identity Manager, Federated Identity Manager, Security Compliance Manager, Security Operations Manager, Directory Server, and Directory Integrator, Tivoli offers a complete set of products designed to address these challenges.

  This book describes the major logical and physical components of each of the Tivoli products. It also depicts several e-business scenarios with different security challenges and requirements. By matching the desired Tivoli security product criteria, this publication describes the appropriate security implementations that meet the targeted requirements.

This book is a valuable resource for security officers, administrators, and architects who want to understand and implement enterprise security following architectural guidelines.

- *Federated Identity Management and Web Services Security with IBM Tivoli Security Solutions*, SG24-6394

  Today, companies have no way to trust identities belonging to their partners, suppliers, contracts, and their outsourcers. This lack of trust means companies end-up creating online identities (and passwords) for all users. This approach is very costly, inefficient, and creates user frustration with multiple accounts and registrations for each Web Site. Federation is the set of business and technology agreements as well as policies that enable companies to optimally pursue business automation goals that best align with their business model, IT policies, security and privacy goals and requirements.

  This book looks closely at the trust infrastructure over which business federations are implemented. We cover important aspects of utilizing the Tivoli integrated identity management architecture in order to build and deploy the Tivoli Federated Identity Management and Web Services Security components, which consist of Tivoli Federated Identity Manager, IBM WebSphere Application Server, and the IBM Integrated Solutions Console.

  This book is a valuable resource for security officers, administrators and architects who wish to understand and implement Web Services security and federated identity management.

- *Federated Identity and Trust Management*, REDP-3678

  This IBM Redpaper™ publication discusses the IBM Tivoli software strategy and roadmap for a consistent, unified, and evolutionary approach to securing cross-enterprise e-business solutions. Built on open security standards, with tight integration to Web middleware, such as Java 2 Platform Enterprise Edition (J2EE™) and Microsoft® .NET, Tivoli security solutions allow you to increase the reach of your business. They build on existing Web security investments that can quickly evolve to take advantage of Web services and federation standards.

- *Propagating Identity in SOA with Tivoli Federated Identity Manager*, REDP-4354

  SOA connects loosely coupled services to construct new applications. These services have their own user registries that are often administered in isolation from those of other services in the SOA environment. Users and service entities in a homogeneous environment are likely to have different identities in the various services that make up a composite application.

  Establishing the identity of the service requester in each service request is a fundamental step in ensuring that business requirements, such as authorization, audit, and compliance, can be implemented. Identity services are required in the SOA infrastructure so that services can be easily interconnected with the correct identities being propagated.

A solution for the challenge of SOA identity propagation must be:

- Capable of understanding and operating with a variety of formats for representing identity
- Capable of translating between different identities
- Based on SOA principles itself to deliver a flexible, infrastructure-based solution de-coupled from application business logic
- Constructed using open standards to provide maximum interoperability with the platforms and systems on which SOA solutions are constructed

IT Architects responsible for designing secure SOA solutions can gain an appreciation for the importance of identity propagation in an SOA and how components of Tivoli Federated

Identity Manager provide an open and flexible solution for identity propagation in SOA. IT Specialists that are required to implement security infrastructure for SOA can learn how to install the Tivoli Federated Identity Manager components that provide a secure SOA identity propagation solution.

# 2 Planning

In this chapter, we provide an overview of IBM Tivoli Federated Identity Manager (TFIM) concepts where identity management is necessary between business partners. Participation in a federation allows a user from one federation business partner to seamlessly access resources of another business partner in a secure and trustworthy manner. These secured transactions are created such that legal agreements, defined in contracts with business partner relationships, are enforced. Tivoli Federated Identity Manager provides functionality adhering to recent standards that provide an effective means for business partners to plan and architect solutions.

In addition to federation concepts, we also discuss federation standards, federated single sign-on (F-SSO), federation services, and Web services security management (WSSM).

## 2.1 Federation concepts

Federated identity technology is used to create a globally interoperable online business identity, driving relationships or affinity-driven business models between companies. The concept is nothing new, because we have real-world models for federated identities of individuals: a passport is a global identity credential that vouches for one's identity in a country; an ATM card is a credential that vouches for one's bank account; a driver's license vouches for one's ability to operate a motor vehicle and is also frequently used as a proof of identity in many business transactions. See Figure 2-1.

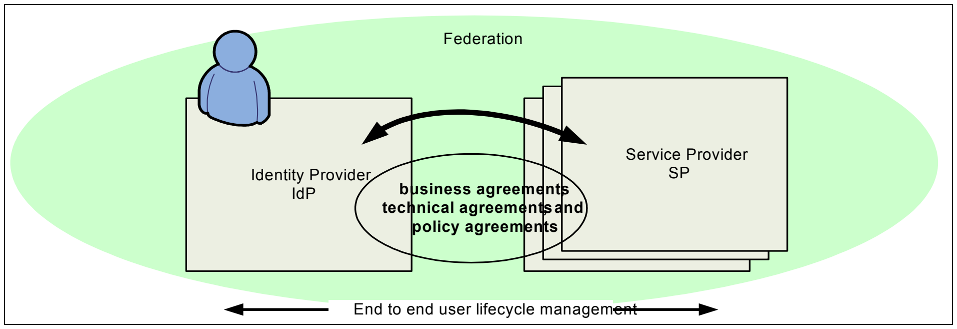


*Figure 2-1 Federated identity management*

*Federated identity management* is based on business agreements, technical agreements, and policy agreements that allow companies to interoperate based on shared identity management. This helps companies to lower their overall identity management costs and provide an improved user experience. It leverages the concept of a portable identity to simplify the administration of users and to manage security and trust in a federated business relationship.

Integration can be simplified because there is a common way to network identities between companies or between applications. Organizations can implement business strategies that drive organic market and customer growth by eliminating the friction caused by incompatible identity and security management between companies.

### 2.1.1 Federation example

The potential benefits of federation and federated identity management are best described by an example. Consider a scenario with the following (fictitious) entities:

- An employer, ABCBigCorp; and an employee, Employee One
- A travel provider, ABCRBTravel

- A service provider, ABCRBTelco
- A bank, ABCRBBanking
- A stock information provider, ABCRBStocks
- A user, John Public, over the Internet

The involved businesses interact with each other, creating a value net of services available to users whether they are public users or employees of a business; see Figure 2-2.

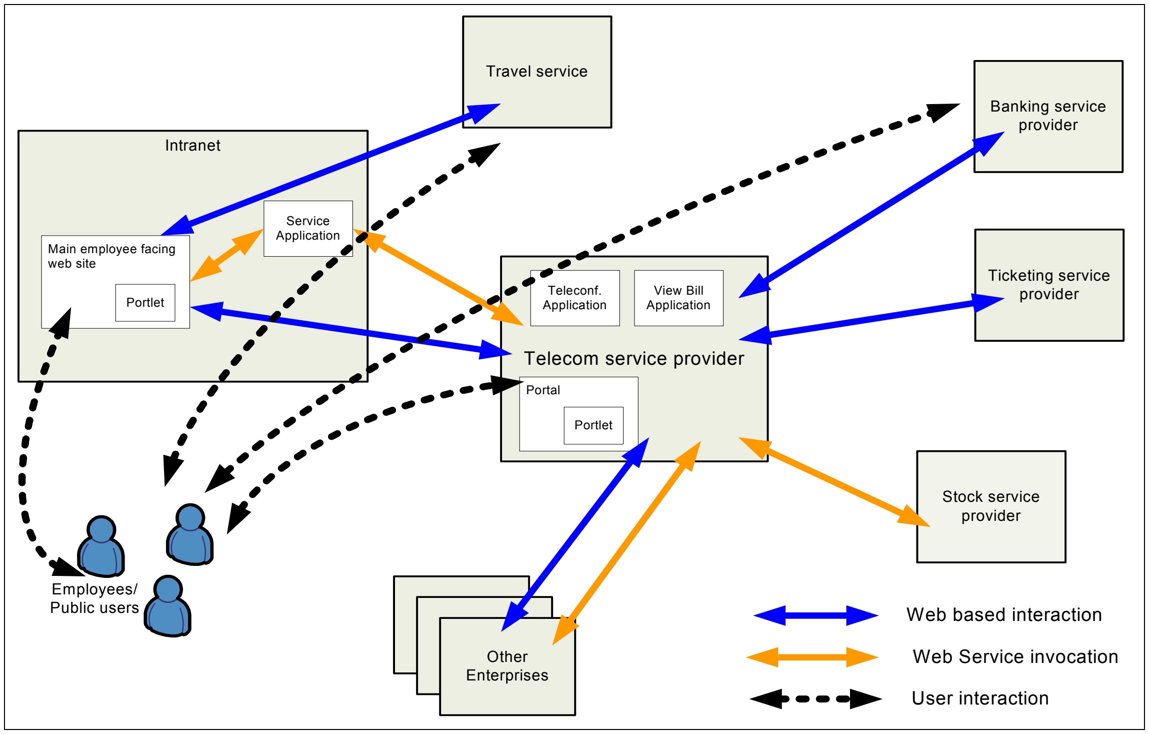


*Figure 2-2 Federation example environment*

The entities in our example are described in the following list:

- ABCBigCorp

  ABCBigCorp is a large company with many employees. As part of providing benefits for its employees, ABCBigCorp provides (subsidizes) health care, retirement savings plans, and other employment-related services such as subsidized mobile phone accounts. As part of reducing its employee costs, ABCBigCorp has outsourced these employee benefits to third-party benefit providers. Because ABCBigCorp is responsible for the management of its users, from account creation (initial hiring) to account deletion or inactivation (dismissal, retirement, and other severance), ABCBigCorp naturally continues to assume this functionality but leverages this in its relationships with third-party benefit providers.

- Employee One

  Employee One is a typical ABCBigCorp employee. The employee has access to the typical services that are provided (brokered) by ABCBigCorp. Employe One also leverages additional services brokered by ABCBigCorp and provided by third-party providers, including travel services, a ABCBigCorp-sponsored mobile phone plan, participation in a stock plan, and online banking.

- ABCRBTravel

  ABCRBTravel manages travel-related services for other businesses, allowing those businesses to order and pay for flights, trains, car rental, hotels, and much more. ABCRBTravel has agreements with the businesses using its service, allowing any employee from the business (and who is directed to the ABCRBTravel Web site) to automatically get an account.

- ABCRBTelco

  ABCRBTelco is a telecommunications service provider. It offers telephony services and also has a portal where ABCRBTelco users or business partner users can choose among offered services to which ABCRBTelco will act as identity provider, offering SSO to the services. ABCRBTelco also has services in its portal, connecting to external service partner Web services and presenting those services in the portal. In addition, ABCRBTelco acts as a service provider to large enterprises, such as ABCBigCorp.

- ABCRBBanking

  ABCRBBanking offers banking services to its own customers directly and also to ABCRBTelco customers through the ABCRBTelco portal.

- ABCRBStocks

  ABCRBStocks offers a stock quote Web service. The service offers several service levels, depending on the user of the service. ABCRBTelco offers this stock service on its portal.

ABCBigCorp is one of the identity providers in these federation relationships. It manages a user registry containing information about all of its employees. ABCBigCorp is responsible for managing the life cycle of its employees, from account creation to account deletion or inactivation.

ABCBigCorp enters into a business federation with a travel services provider, ABCRBTravel. ABCRBTravel is to manage a set of services for all of ABCBigCorp's employees. ABCRBTravel is required to manage information about all of these employees because this information is relevant to

ABCRBTravel's day-to-day management of the employee travel-specific information, such as preferences, frequent flier information, and so on.

Employee One has an account at ABCBigCorp for accessing any ABCBigCorp resources necessary for performing the employee's job. This account is based on being employed at ABCBigCorp. If Employee One goes on a leave of absence, this account can be suspended. If this employee seeks employment elsewhere, this account can be terminated.

Employee One, by virtue of being an ABCBigCorp employee, also has a sponsored account with ABCRBTravel, the travel service company that acts as a third-party service provider to ABCBigCorp. Employee One's account with ABCRBTravel is sponsored: it is created as a direct result of being a ABCBigCorp employee. Employee One is able to access travel information through the ABCBigCorp employee portal. That is, the ABCBigCorp employee portal has a link to ABCRBTravel's Web portal that redirects Employee One from ABCBigCorp to ABCRBTravel in order to access Employee One's externally available services and information.

Without federation, Employee One has to explicitly authenticate to the ABCRBTravel site to access the account, even though Employee One has already authenticated to ABCBigCorp and has accessed the ABCRBTravel's Web site services through Employee One's portal.

By entering into a federation relationship, ABCRBTravel can reduce its overall cost of managing users. The bulk of this process is achieved by participating in single sign-on and no longer directly managing Employee One's authentication credentials, which, by many reports, is an expensive part of user life cycle management.

From Employee One's point of view, having ABCRBTravel and ABCBigCorp participate in a federation relationship with reduced sign-on allows Employee One to authenticate once to ABCBigCorp and then access his travel information without having to explicitly re-authenticate. This is achieved with federated reduced sign-on.

Federated reduced sign-on between an issuing domain (ABCBigCorp) and a relying domain (the federated service provider ABCRBTravel) facilitates the secure and trusted transfer of user identifiers and other attribute-related information (such as authorization roles, group memberships, user entitlements, and user attributes such as Employee ID and credit card number).

What is required is that ABCRBTravel is able to participate in a runtime exchange of information with ABCBigCorp which results in some assertion from ABCBigCorp (note that this exchange of information requires no interaction with Employee One). This assertion is then trusted by ABCRBTravel and used to uniquely identify Employee One based on an ABCBigCorp asserted unique identifier. Using this information, ABCRBTravel is able to locally identify and provide access to Employee One's benefits account information.

Note that both ABCBigCorp and ABCRBTravel need to maintain information about Employee One. There will be attributes about Employee One that are best managed by ABCBigCorp, such as Employee One's home address and telephone number. Likewise, there will be information about Employee One's travel preferences that are clearly not appropriate for ABCBigCorp to manage on behalf of ABCRBTravel. Thus, ABCRBTravel is able to personalize a user's experience based on ABCRBTravel-maintained attributes.

The second major player in this example is ABCRBTelco. ABCRBTelco offers services to businesses and public users. When offering services to businesses, it does not necessarily care about the individual employee at the business but will treat them all as one user with regards to authentication. Offering the ability to book teleconferences might be a service that is available only to businesses. Attributes that are forwarded from businesses would allow ABCRBTelco to personalize the user experience further if necessary.

Public users to the ABCRBTelco portal have a personal account. Public users who are customers of ABCRBTelco benefit from its partner service offerings presented by the portal. The services would allow for reduced sign-on, enabling the user to log on only to the ABCRBTelco portal. Then, the user can select that service by clicking on the link in the portal and then connecting to the offered services without having to log on again. One such service is the ABCRBBanking, offering its customers access to its bank services through the ABCRBTelco portal with reduced sign-on, in the same way as ABCRBTravel offered its services to ABCBigCorp employees.

Some services at the telecommunications portal are consumed Web services from partners to ABCRBTelco. Web services are not accessed by the user being redirected to another Web site and benefitting from reduced sign-on, but instead Web services is accessed by a local ABCRBTelco application. The ABCRBTelco application benefits from the end-to-end security offered by the Web services security interaction

Information about the user, and that is necessary for the stock application to be able to deliver the quality of service-based information relevant to the user's credentials at ABCRBTelco, is included in the request from ABCRBTelco.

## 2.1.2 Federated identity management architecture

*Federated identity management* (FIM) functionality enables companies and business partners to lower their overall identity management costs, improve user experience, reduce the company pain points, and mitigate security risks for transactions.

When discussing identity federation, the following solution areas (also shown in Figure 2-3) are defined:

- Web-based single sign-on: Federated single sign-on referred to as F-SSO
- Application-based Web services security: Secure Web services referred to as Web services security management (WSSM)
- Identity life cycle: Federated provisioning

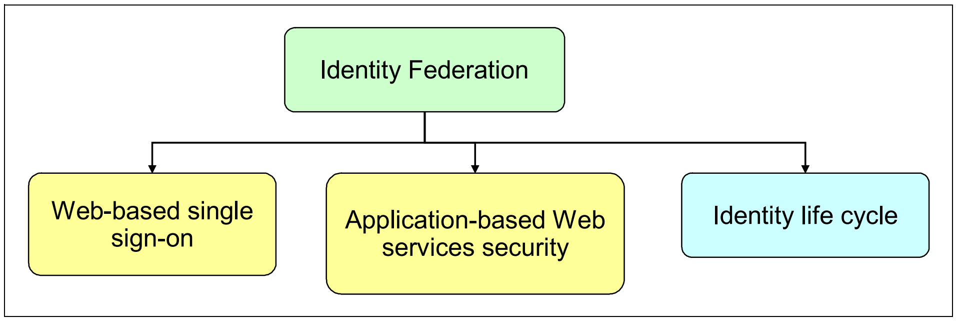


*Figure 2-3 Federated identity management solution areas*

In the next sections, we cover the common fundamentals and terminology for the three solution areas, starting with a background on FIM, an architecture overview, and finishing with the general architectural FIM terminology and concepts.

## 2.1.3 Background to federation

Federation solutions are successful when they allow customers, business partners, and users to integrate easily between the federation business partners without having to constantly manage security and identity in the process in a per relationship proprietary way. Unfortunately, current implementations for managing security and identity data often force users and businesses to manually manage access, trust, transport and identity attributes. Often this burden has a heavy impact on both ability to execute and on growing administrative cost because each business has to administer a large and rapidly changing base of identities. Such a model is an impediment to the adoption of federations, and is a pain-point for both users and businesses.

Federation technology is used to:

- Provide a simple mechanism to identify and validate users from business partner organizations and provide them with seamless access to Web sites within that trusted federation.
- Support standards-based end-to-end trust and security for applications exposed as Web services between businesses.
- Offload the expensive part of the user management (the cost of user enrollment, account creation, password management and user care) to one business partner (an identity provider).
- Standardize the provisioning of users and attributes to support both user and application based interactions, extending enterprise identity management to inter enterprise identity management
- Reduce the need of business partners to manage large sets of user data, including the cost of managing authentication credentials for large numbers of users.

The goal of federation is to support a dynamic and seamless integration of services and resources between businesses within a federation.

An organization typically is willing to pursue a federation model when they can rationalize the benefits of such a solution against the risks of supporting a business model based fundamentally on third-party trust. An organization will have extreme difficulty to engage in a federated model when it does not have the same visibility of life cycle management of third-party users as it does with its direct users. Therefore, federated identity life cycle management is an approach to deliver the same kind of visibility around an identity-related business process when organizations begin to loosely couple very disparate identity management systems across trust domains.

One of the most pressing issues for an IT administrator is how to implement the technical policies and operational best practices; how to implement and enforce security and identity agreements, audit and privacy agreements, so that the federated relationships appear as an extension of the existing identity management procedures.

### 2.1.4 Architecture overview

Federated relationships can be based on proprietary technologies that allow business partners to communicate and collaborate. In general, a proprietary approach is not scalable or maintainable across a large set of partners. For this reason, standards and specification-based approaches are rapidly gaining in popularity. Federations facilitate an integrated approach to business. Federations are entered in to facilitate two major types of functions:

- Seamless and secure user interaction across federation business partners (also known as *federated single sign-on*, or *F-SSO*)
- Seamless and secure business interaction across application platform integration (also known as *Web services security management* for service-oriented architectures)

Both of these types leverage the same basic functionality, namely, both require a *trust infrastructure*. The trust infrastructure provides the technical representation and implementation of the business and legal agreements between business partners, as shown in Figure 2-4. Both F-SSO and Web services security solutions are built on a trust infrastructure.

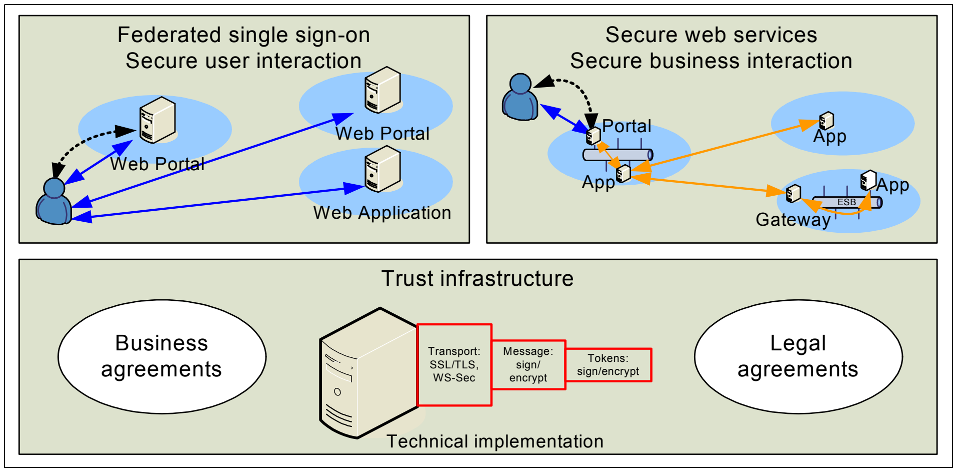


*Figure 2-4 Base trust infrastructure for secure services*

Federated identity management often refers to user-driven, browser-based interaction between organizations. This space is reference to as *federated single sign-on* (F-SSO) even though it has largely matured beyond simply single sign-on functionality. Standards and specifications such as the SAML specification and WS-Federation and Liberty Alliance ID-FF (Identity Federation Framework) specifications all now include an aspect of session life cycle management (single sign-on and single sign-off) as well as single sign-on enablement through account linking. This comprehensive approach and enablement of a single sign-on environment is designed to ease the user experience and reduce the cost of management of these users. For example, previously a user had to establish an account, including user name and password, at each business partner; the business partner in turn had to assume the cost of managing this user and the user's access to the user's system.

Federation solutions ease this cost by reducing the amount of information that must be managed for each user and the overall cost of managing this information.

As Web services evolve, currently boosted by the industries' drive towards building service-oriented architectures, the need to expose them to external businesses will increase rapidly. Web services security targets the secure interoperability of applications or programs. Web services provide a flexible and easily adoptable means of integrating applications. Web services security defines how to do this in a secure manner. This approach includes securing the message through signatures and encryption. It also includes authenticating and authorizing requests based on the Web services invoker's claimed identity. This identity is represented with a Web services security token. This process of authenticating a principal's identity (user or application) is a form of reduced-sign-on.

Unlike the federated identity management single sign-on we described, however, this process occurs in what is often referred to as an *active client* environment. This means that the applications that are invoking Web services are able to assert their claimed identities in a Web services request without having to negotiate a separate (dedicated) single-sign-on protocol.

IBM provides the necessary functionality to implement the trust infrastructure used by both of these solutions; this functionality is provided by a *trust service*. Layered over the trust service functionality are two largely independent but complementary solutions: *Federated single sign-on (F-SSO)* and *Web services security management*.

To design a solution, you should understand the following areas, which are covered in this section:

- The roles of identity and service provider

  The definition of who is the authoritative source of the user identity information

- Identity or attribute mapping

  The definition of the attributes to be shared and the mapping of them in the business partner systems

- User account management or provisioning

  The procedures for managing user identity data agree what information can be shared, what information is independently managed by users, and whether the users will be provisioned automatically to the new endpoint (prior to or at runtime)

- Account linking

  The procedures for managing the account linking, to agree on some common unique identifier for the user, which can be bounded with the internal, local user identity at the service provider. This step also involves the definition of the account de-linking and de-provisioning procedures
- Trust

  The process of ensuring security for connections and transport, messages and tokens
- Selection of the federation protocol profile or profiles

  The definition of the federation protocol profiles to be used between the two business partners

## 2.1.5 Roles

Within a federation, business partners play either or both of the following roles: *identity provider* or *service provider*. The identity provider (IdP) is the authoritative site responsible for authenticating an user and asserting an identity for that user in a trusted fashion to trusted business partners. Those business partners who offer services but do not act as identity providers are known as service providers. See Figure 2-5. The identity provider takes on the bulk of the user's life cycle management issues. The service provider (SP) relies on the IdP to assert information about a user, leaving the SP to manage only those user attributes that are relevant to the SP.

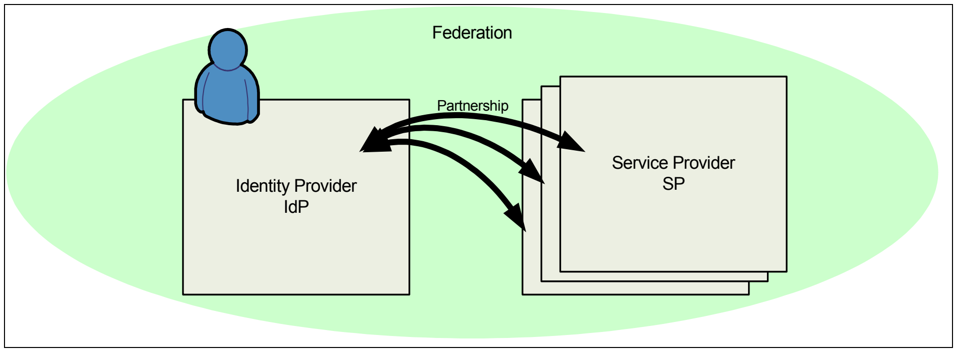


*Figure 2-5 Identity provider and service provider in the federated model*

### Identity provider

The identity provider (IdP) is responsible for account creation, provisioning, password management, and general account management, and also acts as a collection point or a client to trusted identity providers. Having one federation business partner act as a user's IdP relieves the remaining business partners of the burden of managing equivalent data for the user. These non-IdP business partners act as service providers (SPs). The SPs leverage their trust relationships with an IdP to accept and trust vouch-for information that is provided by an IdP on behalf of a user, without the direct involvement of the user. This enables businesses (service providers) to offload identity and access management costs to business partners within the federation.

To achieve the overall user life cycle management required for a full federated identity management solution, the identity provider assumes the management of *user account creation*, *account provisioning*, *password management*, and *identity assertion*. The identity provider and service provider cooperate to provide a rich user experience by leveraging distinct federated identity management profiles that together provide a seamless federation functionality for a user.

#### Service provider

A service provider (SP) may still manage local information for a user, even within the context of a federation. For example, entering into a federated identity management relationship might allow a service provider to handle account management (including password management) for an IdP in addition to the management of its user-specific data (for example, SP-side service-specific attributes and personalization-related information). In general, a service provider offloads identity management for an identity provider to minimize its identity management requirements while still enabling full service provider functionality.

### 2.1.6 Identity attributes

In a federated model, an identity provider and service provider have to agree on what information they can share with respect to a user identity and what information must be independently managed. This information is composed of classes of data that concern an identity (see Figure 2-6 on page 39), as follows:

- Authentication credentials
- Transaction attributes
- Profile attributes
- Provider-specific attributes

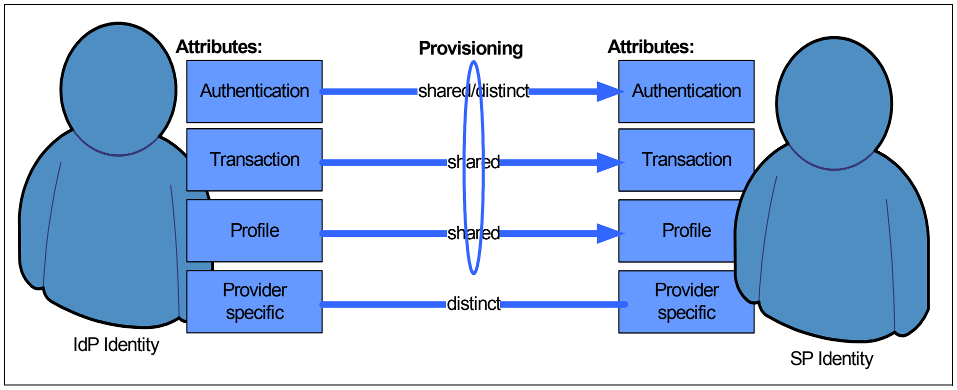


*Figure 2-6 Shared and distinct identity data and attributes*

For each class of *identity data*, we can allow for a *shared* or *distinct* identity data management solution as shown in Figure 2-6. Thus when examining the provisioning requirements for a federated model, we evaluate the shared/distinct nature of each of the classes of identity data.

### Authentication credentials

Authentication credentials are the information used to authenticate an identity. This information is bound to a user's identifier (such as a user name or logon identifier). The authentication credentials themselves are represented by data such as a password or a one-time-generated PIN number from a hardware token. These credentials are presented by a user as part of the authentication process and used to prove (authenticate) the user's claimed identity. This implies that to authenticate a user, a federation business partner must have a copy of the user's authentication credentials, or another means of validating the user's authentication credentials. Thus, current models of authentication require a distinct identity data model, meaning that each business partner has a copy of the user's authentication credentials.

One goal of a federated model is to move to a shared identity data model. With authentication credentials, this implies that a federation business partner be able to trust a third party (an identity provider) to evaluate the user's authentication credentials and to assert some form of secure, trusted information that can be used to vouch for the user's successful authentication at the identity provider. Therefore, in a federated model, authentication credentials may be extended to include security tokens from an identity provider asserting the user's identity.

Moving to a shared model for authentication credentials means that federation business partners are able to act as service providers and no longer have to manage the class of identity data, including authentication credentials.

Provisioning solutions are used to tie the identity account management at an identity provider to that at a service provider.

A shared identity approach to federated business interactions can be appropriate when one business partner is able to trust and rely on the assertion of a user's identity by an identity provider without having to independently validate the user's authentication credentials. In this model, federation allows the user (and the federation business partners) to establish a common unique identifier to use to refer to the user, where this identifier reveals no information about the user at either business partner. Based on this common identifier, an identity provider is able to issue single sign-on information to federation business partners.

In a shared identity model, provisioning authentication credentials is not necessary. There is, however, a need to somehow establish a user's local identity and the common identifier used by the two business partners. This is handled through a provisioning solution. In general, a distinct identity account data model does not involve a provisioning solution. The user in this federation model has distinct identity accounts at both of these business partners, maintained and administered independently at both the identity provider and service provider. With a distinct identity data management model, identity data may be initially provisioned across business partners as part of the initial account setup, although it will be managed independently (outside the scope of a provisioning solution) after this.

In certain cases, this approach might not be true, for example, if the user does not already have a distinct, authenticable account at both the identity provider and the service provider. In this case, the identity provider may trigger a provisioning event at a business partner to create a local identity account and identity account data for a user. Part of this action may including establishing a common identifier used by the two business partners. As with the shared data approach, provisioning solutions, when invoked within a distinct identity model, can be one of two types: prior to or at runtime provisioning.

### Transactional attributes

Transactional attributes include information that describes a user and the user’s affiliations and entitlements. This information is bound to a user's identifier. This information can include groups that the user belongs to or roles that the user can assume. It can also include additional identifiers (such as customer ID number, 401K account number, frequent flier status level, health care number, supplier ID, or billing or credit card number, and so on), specific organizational roles (such as HR manager, stock broker, benefits administrator, primary care physician, executive, supervisor, travel exception approver, and so on).

This information is often used as part of access control decisions at the transactional level (for example, can this HR manager update this employee's personnel evaluation?). This information about a user is not normally managed by the user. In general, a user's transaction attributes are not common across all identity and service providers; not all of these attributes are relevant to all identity or service providers.

Sharing of transactional attributes allows one of the parties (usually the identity provider) to act as the *authoritative source* of transactional attribute information about a user. This attribute information can then be provisioned to a service provider in an *a priori* manner, meaning that when this information is updated at the identity provider, an a priori provisioning request will attempt to update this information at the service provider. This attribute information can also be provisioned in a dynamic, or just-in-time, manner, meaning that updated or new information is included as part of a single sign-on response to the service provider, or in response to a direct request from the service provider.

When transactional attributes are distinctly managed within a federation, each federation business partner is responsible for the day-to-day management of these attributes. This means that a provisioning solution is not implemented as part of the day-to-day management of these attributes. With a distinct identity data management model, transactional attributes may be initially provisioned across business partners as part of the initial account setup, although it will be managed independently (outside the scope of a provisioning solution) after this. Note that because transactional attributes are typically not managed by the user, this day-to-day management must be handled by the service provider's administrators.

### Profile attributes

Profile attributes represent auxiliary information that is not primarily tied to authentication or authorization decisions. Profile attributes may be information that is specific to the user identity, such as e-mail address, home address, birth date, and telephone number. Identity profile attributes also include preference or personalization attributes such as a user's frequent flier number, location information, and preferences and subscription information (for example, user subscribes to a newspaper, and so on). This information may be used as part of secondary user identity validation (as part of a lost password-recovery process).

This information may be used as part of an access control decision in scenarios where access is controlled by, for example, a user's age or residential address. This information about a user is normally managed by a user. In general, a user's profile attributes are consistent across identity and service providers.

To put this into a familiar context, consider a the ABCBigCorp employee, Employee One, who participates in a frequent-flier program with his airline of choice. Employee One has an online travel account at ABCRBTravel and uses it to book air travel; this account is bound to Employee One’s identity. Associated with this user name is Employee One's password (authentication credentials) used to authenticate. These credentials are not known by Employee One because they were set up as part of the provisioning from ABCBigCorp. Associated with Employee One's travel account are Employee One's profile attributes (for example, billing address, e-mail, and telephone number).

Based on Employee One's travel account, the travel service assigns and manages Employee One's frequent-flier status (a transactional attribute). When attempting to book a flight, Employee Ones attributes will be used to assist in booking the flight and also enable the ticket to be issued to Employee One's frequent-flier card. When Employee One attempts to book a trip, the travel class might be based on attributes with regards to Employee One's airline points or position at ABCBigCorp. After Employee One has selected a desired travel plan and is about to book it, a secondary evaluation of Employee One's identity will be accomplished as part of the specification of Employee One's billing address (to which the ticket confirmation information is to be sent).

Provisioning solutions allow the identity provider to create or update user profile attribute information such as e-mail, personal information, address, membership or subscriber information, and service-specific attributes about a user to service providers. These attributes are typically managed *by the user* (by managing profile information at the user's identity provider).

#### Provider-specific attributes

Provider-specific attributes include both transactional and profile attributes that are relevant for a given user at a given service provider; these attributes have not been shared with other service providers. Examples of provider-specific transactional attributes can include a user's buying history maintained with an online auction house and the bonuses (free shipping) associated with this user's transaction history. Examples of provider-specific profile attributes can include a user's preference to always search for new auction items within the "Toys less than $25" category, for example.

A user's provider-specific attributes are just that: They are distinct attributes that are not shared across federation business partners and are not required to be managed through a provisioning solution across business partners.

### 2.1.7 Trust

*Trust* is a key capability for all three solution areas, and therefore a key area for federated identity management (FIM).

A *trust relationship* is represented at a technical level by cryptographic keys that are used to sign and encrypt messages. These types of cryptographic techniques provide a trust infrastructure over which other services can be layered.

To help ensure a desirable user experience, business partners within a federation have to communicate information about a user in a secure and trusted fashion. This is accomplished by leveraging a trust infrastructure (Figure 2-7).

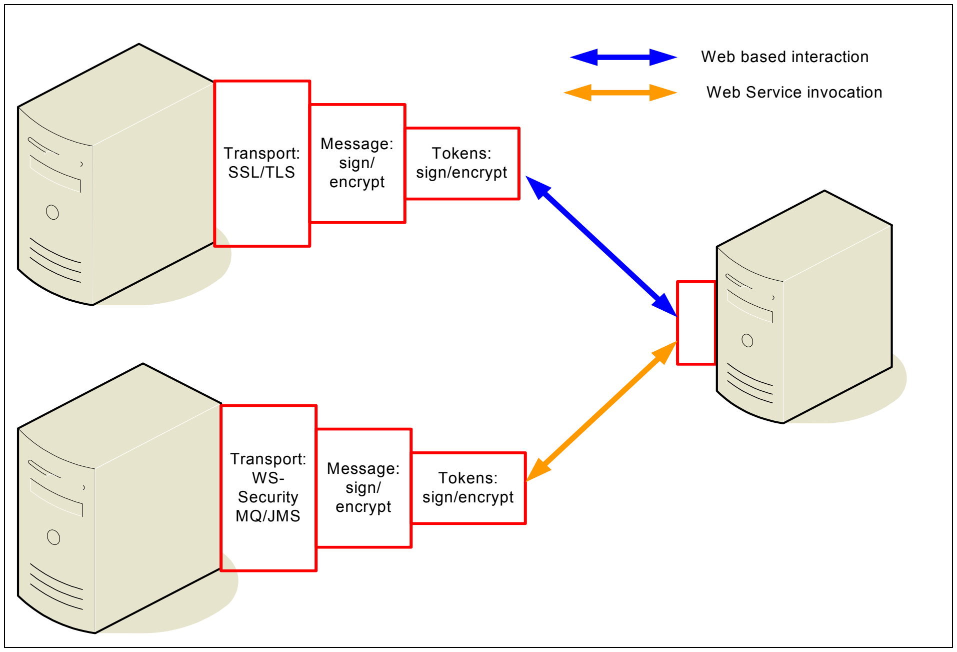


*Figure 2-7 Layers of trust*

A trust infrastructure enables the protection of a message at all levels:

- Transport

  Using SSL to protect user-based FIM communications or WS-Security to protect application based FIM communications

- Message

  Using encryption and signing to provide confidentiality and integrity protection on messages within a FIM flow

- Token

  Using secure tokens to communicate information about a user as part of specific steps within the FIM flow

The trust infrastructure provides protection against invalid or fraudulent FIM flows and allows for a single point of management of the trust information.

### Transport

The simplest form of trust infrastructure is provided by the transport layer SSL protocol, which is used to encrypt communications at the transport layer, between two business partners. Enterprises generally understand how to manage SSL certificates and how to use them to authenticate other enterprises with techniques such as mutually authenticated SSL. SSL-based trust infrastructures suffer from some limitations, notably that they are (at best) point-to-point based, not end-to-end.

Web services, however, might not always run over SSL-compatible transport protocols; Web services can be invoked with transport layer protocols such as JMS or MQ. Thus a Web services trust infrastructure requires more flexibility than offered by SSL. This flexibility is provided by encryption and signing of Web services requests themselves in addition to any transport-level protection that might be applied.

Federated identity management requests usually run over HTTP and thus be able to take advantage of SSL. They are not point-to-point communications, however, meaning that an additional layer of protection is required. This is provided by encryption and signing of the FIM requests themselves in addition to any transport level protection that might be applied.

### Message

For both Web services and federated identity management solutions, a non-transport-based trust infrastructure is required. This is provided by the use of signing and encryption of requests at the *message* layer. The trust service provides the infrastructure to manage the keys and certificates used for this signing and encryption.

The *trust service* provides a means of managing one's own keys and certificates, and of binding a business partner's certificates (validated by a third-party certificate authority) to the local, business-agreement validated, business partner identity. These keys and certificates are then used to sign/validate, and encrypt/decrypt messages between business partners, independent of any transport layer security.

### Token

In addition to message layer security, security *tokens* may be included in a message to convey security-specific information (used for authentication or authorization purposes, for example) about a requestor. This information is part of the trust infrastructure in the same way that keys are used for signing and encryption purposes: The proper use of these tokens conveys information about the holder of these tokens.

The trust service provides a means of managing these security tokens. These tokens are common to (at least) one other business partner and contain pre-arranged security-relevant information. These tokens are themselves protected through signing and encryption, often using the same keying material as used at the message layer.

### 2.1.8 Federation protocol

When creating a federation, an agreement has to be made on a technical level of what FIM standard to use within the federation. An identity provider will most likely support several, and even service providers might do the same, but one has to be defined for each federation partnership.

The different identity standards and efforts in this space, and their dependant standards are discussed in 2.2, “Federation standards” on page 45. The various standards have different capabilities that govern the choice of protocol.

## 2.2 Federation standards

Federation standards depend, not only, on descriptions of XML structure, but also depend on existing transport and data manipulation standards. We discuss these standards in this section. We also discuss identity standards supported by Tivoli Federated Identity Manager.

Federation identity standards depend on XML specification in combination with other standards to define robust set of federation standards. These dependant standards include SOAP, SSL/TLS, and XML signatures, which in combination provide message transportation, transport security, and message security respectively. XSL provides XML manipulation, which allows broad transformation of XML data. XSL is important in understanding how an identity XML structure can be changed, for example, to include various attributes. We discuss these standards in the following sections:

- 2.2.1, “SOAP” on page 46
- 2.2.2, “SSL/TLS” on page 47
- 2.2.3, “XML digital signature/XML encryption” on page 47
- 2.2.4, “XSL” on page 50

Tivoli Federated Identity Manager by design supports the most popular identity standards that are already drafted. These standards are discussed in the following sections:

- 2.2.5, “WS-Federation” on page 52
- 2.2.6, “WS-Security” on page 56
- 2.2.7, “Security Assertion Markup Language” on page 56

**Important:** The standards discussed in this section are specific to data transport, securing of XML data, XML transformation, and federated identity standards. Although browser-related HTTP standards are not discussed, they are still important in understanding the overall data flows, which are affected when enforcing various identity standards.

### 2.2.1 SOAP

SOAP addresses the need in standardizing the method for transferring data between applications. SOAP depends on XML specification, therefore, defines how data must be structured within XML context. SOAP does not address underlying network complexities and does not describe the securing of payload. SOAP uses the HTTP standard as a means to handle network transport of SOAP payloads. The framework is simple and defines the following grammar rules:

- <Envelope>

  This element is required within a SOAP message and may contain namespace declarations. Additional attributes may be also be present but they must be namespace-qualified. Additional supplements are required to be namespace-qualified and located following the SOAP <Body> element.

- <Header>

  This element may be present within a SOAP message. If present, the element is required to be the first immediate child element of an <Envelope> element. The <Header> element may contain header entries. Each entry must be an immediate child element of the <Header> element and must be namespace-qualified.

- <Body>

  This element is required within a SOAP message and must follow the <Header> element. If Header is not present, then <Body> must be an immediate child of <Envelope>. The <Body> element may contain body entries. Each entry must be an immediate child element of the <Body> element and may be namespace-qualified. Fault elements, which contain error messages, may be present within <Body> element.

Many tutorials and references exist describing SOAP in more detail. The standard is described and available at:

`http://www.w3.org/TR/soap/`

**Note:** SOAP standard has recently drafted 1.2 version. For purposes of this section, we refer only to version 1.1.

### 2.2.2 SSL/TLS

Secure Sockets Layer (SSL, standardized as Transport Layer Security, TLS) provides session-level security through the use of encryption. Although not often thought of as an identity management protocol, SSL can be used to authenticate senders and receivers through digital certificates, verify data integrity, and ensure confidentiality. As such, SSL is often the first (and only) option considered in securing transactions over the Internet. It can be used in both browser-to-Web server and server-to-server communications.

Despite its popularity, SSL has shortcomings in the following areas:

- Granularity

  Either all the data over the session is encrypted or none is. This can impact throughput in cases where large amounts of data are exchanged but only small portions actually have to incur the overhead of encryption/decryption.

- End-to-end

  SSL protection ends if intermediate components must examine transactions. No provision is made for encrypting end-to-end across intervening components.

### 2.2.3 XML digital signature/XML encryption

To provide message level security, signature and encryption algorithms can be used individually or in tandem to create a secured trust relationship. Message level security is important when intermediaries are involved during flow of data.

A good example is during a credit card transaction:

- In the purchasing of an item, a merchant only has to know what item an individual wants to purchase.
- In finalizing the purchase, however, the merchant only has to know whether the bank will allow the funds to be transferred from bank to merchant.

Therefore, the merchant does not have to know private account information (such as account balance and account ID). XML encryption fills this gap, so that portions of the data can be encrypted (such as message privacy). This solution allows XML data meant for two or more parties to be contained within a single document or message. XML encryption standard specifies a process for encrypting data, and representing the result in XML. For more details about XML encryption standard, go to:

`http://www.w3.org/TR/xmlenc-core/`

The digital signature concept is not new and has been around for quite some time. In general, digital signatures encompasses three algorithms:

- Public/private key-pair generator
- Signing algorithm
- Signature verify algorithm

Although digital signatures are used widely in SSL/TLS protocol, XML digital signature expands the concept for XML data. The purpose of XML digital signatures is to ensure that XML payload has not been tampered while in transit (sometimes referred to as message integrity). From this requirement, the XML digital standard exists today. Refer to the standard, at:

`http://www.w3.org/TR/xmldsig-core/`

Let us look more closely at the XML digital signature elements:

- <Signature>

  This element is the root element and denotes the beginning of signature description and its related data.

- <SignedInfo>

  This element includes canonicalization and signature algorithms. In addition, one or more references may be present:

  – <CanonicalizationMethod>

    This element describes the algorithm that is applied to canonize the <SignedInfo> element. Canonicalization as defined, is a method to normalize the structure of data without losing initial data. In this case, we refer to XML data. XML data sets that contain same XML information can in fact have differing textual representation. An example is when two XML documents are equal, in respect to elements, attributes and data, but one XML document might contain extra white spaces, which in XML the specification are ignored. Canonicalization comes into play and is necessary when signing or encrypting XML data. The addition of white spaces in XML data can affect the signature value. Therefore, <CanonicalizationMethod> describes an algorithm so that verification of the signature will always be consistent.

  - <SignatureMethod>

    This element is required and specifies the algorithm used for signature generation and validation.

  - <Reference>

    This element may occur multiple times. This element describes a digest algorithm, described in <DigestMethod> element; and a digest value, provided within <DigestValue> element. Optionally, before digest is executed, an identifier of the object being signed, type of object, and array of transforms applied, can be included within the <Reference> element. the <Transforms> element may exist within the <Reference> element and may contain an array of <Transform> elements.

    Additional details of how <Reference>, <Transforms>, <DigestMethod>, <DigestValue> and <Transform> elements can be used, are found in the XML digital signature standard.

- <SignatureValue>

  This element contains the actual value of the digital signature. This value must be always be encoded using base64.

- <KeyInfo>

  This element is optional, and allows the recipients to obtain the key required to validate the signature. <KeyInfo> may contain additional data related to keys, names, certificates and other public key management information.

In Example 2-1, we show an XML digital signature of XML data.

*Example 2-1 XML digital signature example*

```
<ds:Signature Id="uuid2adb9d23-0105-e44f-c899-8ce3efd72411">
               <ds:SignedInfo>
                  <ds:CanonicalizationMethod 
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></ds:CanonicalizationMethod
>
                  <ds:SignatureMethod 
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"></ds:SignatureMethod>
                  <ds:Reference 
URI="#Assertion-uuid2adb9cee-0105-fe74-40da-8ce3efd72411">
                     <ds:Transforms>
                        <ds:Transform 
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"></ds:Transfor
m>
                        <ds:Transform 
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                           <xc14n:InclusiveNamespaces 
xmlns:xc14n="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="saml 
ds"></xc14n:InclusiveNamespaces>
```

```
                        </ds:Transform>
                     </ds:Transforms>
                     <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>

<ds:DigestValue>XJieAD/CpXPPw3q6wnOu2iOLwsA=</ds:DigestValue>
                  </ds:Reference>
               </ds:SignedInfo>

<ds:SignatureValue>BYV5yZ8QY3b8aKm9zaQqmrGIWFYaLwcDUEr5sp7Bgn4i2c/SEk2DErT2zOdW
/nZR2i7uhQ1OZDfu2PrB/ruv3kyMJUVyuy2wHD2Ro4SgQ4kYbxyg6GROtzJC2Cx+EfQz4aioIbV7eKO
LF+NZOhBj2kpb/8TobqTzgOK9L8O3UkE=</ds:SignatureValue>
               <ds:KeyInfo>
                  <ds:X509Data>

<ds:X509Certificate>MIICqjCCAhOgAwIBAgIBAzANBgkqhkiG9w0BAQQFADA5MRwwGgYDVQQDExN
maW0ucmVkYm9vay5pYm0uY29tMQswCQYDVQQGEwJVUzEMMAoGA1UEChMDSUJNMB4XDTA1MDYwMTIxMj
QzOVoXDTEwMDYxNjIxMjQzOVowRjElMCMGA1UEAxQcYmlnY29ycF9yYnRyYXZlbC5iaWdjb3JwLmNvb
TELMAkGA1UEBhMCVVMxEDAOBgNVBAoTB0JpZ0NvcnAwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGB
AL+up7hIOvMJB/g9ZhglKTW3x/PxTVhG5l6hJ3kNdrZeJhPg59usfWmrEJSn2UglEGSGz5kTWvYS2dn
AMANDAoWESTMANgbyzLdpOb2iLKsLekyRcRk+u6i6Hbs8gOzoLGJyv+zZaOLSUyOj186SrGb8L575PA
Ws5jlkwPlULohPAgMBAAGjgbQwgbEwDAYDVR0TAQH/BAIwADAdBgNVHQ4EFgQUWW6uNP9izCSYZO4Tt
eb6Sa9bx2owYQYDVR0jBFowWIAUQNM+O+Jvv8jfpobQbQhsXg/LkTGhPaQ7MDkxHDAaBgNVBAMTE2Zp
bS5yZWRib29rLmlibS5jb20xCzAJBgNVBAYTAlVTMQwwCgYDVQQKEwNJQk2CAQEwCwYDVR0PBAQDAgS
MARTINGgWESTMANiBDQQFFgNocGgwDQYJKoZIhvcNAQEEBQADgYEANmvniu+bM1gS3iBSK1w/3X/rZL
3GTPOoMlUCcGhzy1mFOxKe+Bm/lIaqG2qdx2uGjKkeOACjecNM93Je9PfYb7XP1p53C7azCOZIsOeiw
fTRDShWtQqQoOwduIYafJSeQNn14zakIxCReVSKUXs2eQdBCLi4KVxHN8Zg6W1xwdQ=</ds:X509Cer
tificate>
                  </ds:X509Data>
               </ds:KeyInfo>
            </ds:Signature>
```

### 2.2.4 XSL

Extensible Stylesheet Language (XSL) is a family consisting of transform and format languages. XSL standard and details can be found at:

http://www.w3.org/Style/XSL/

The following languages are currently defined:

- XSL Transformations
- XML Path Language
- XSL Formatting Objects (XSL-FO)

In this book, we discuss XSLT and XPath, but not XSL-FO.

## XSL Transformations

The XSL Transformations (XSLT) standard can be found at:

http://www.w3.org/TR/xslt

As the name implies, XSLT is an XML-based language that is used to transform XML data. The specification provides a language basis, with syntax and semantics that can then be followed to form the transform code. XSLT contains a set of template rules, which has two parts: pattern matching, against nodes within XML source tree; and a template, which can be instantiated to form the resulting tree. The resulting tree is the basis of the XLST processor output. The flow of the transformation can be seen in Figure 2-8.

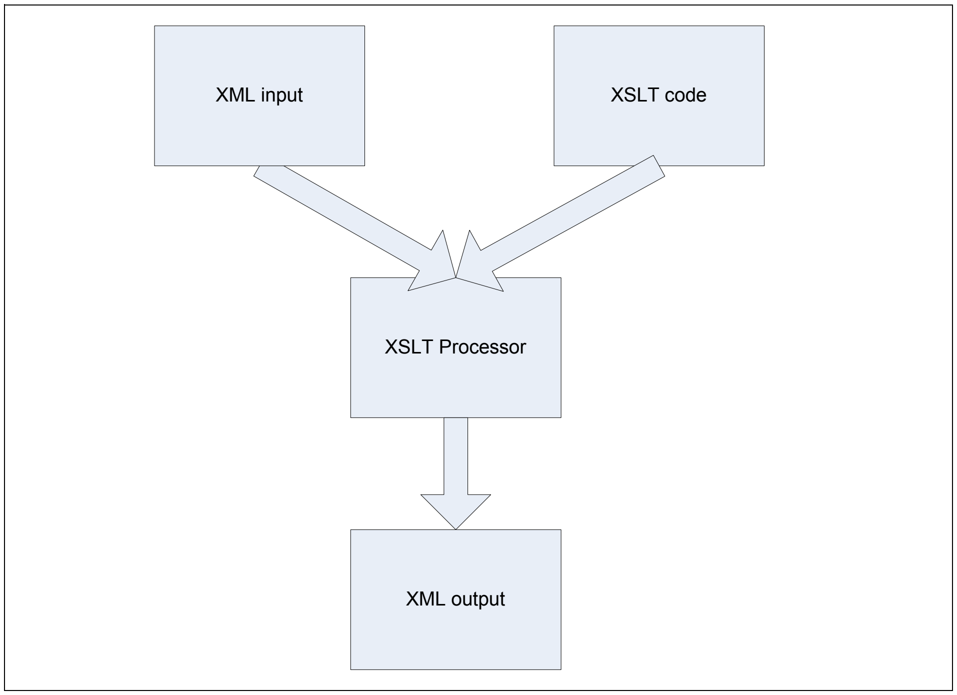


*Figure 2-8 Flow of XML transformation*

For an introduction to XSL/XSLT, refer to the tutorial at:

http://www.w3schools.com/xsl/default.asp

Several constructs of the XSLT language include:

```
<xsl:template>
<xsl:value-of>
<xsl:for-each>
<xsl:sort>
<xsl:if>
<xsl:choose>
```

#### XML Path Language

XSLT depends on XML Path Language (XPath) to provide common syntax and semantics, primarily used to identify select nodes or subset of nodes, within a XML source document. XPath also provides facilities for string, number and Boolean manipulation. A simple XPath model closely resembles URI semantics. XPath standard can be found at:

`http://www.w3.org/TR/xpath`

For information about XPath, refer to the tutorial at:

`http://www.w3schools.com/XPath/default.asp`

### 2.2.5 WS-Federation

WS-Federation describes how to use existing Web services security building blocks to provide federation functionality, including *trust*, *single sign-on* (and *single logout*), and attribute management across a federation. WS-Federation is really a family of three specifications, which are *WS-Federation*, *WS-Federation: Active Requestor Profile*, and *WS-Federation: Passive Requestor Profile*, as follows:

- *WS-Federation* itself describes how to implement a federation in a Web services world. In particular, WS-Federation focuses on the relationships between parties, and the high-level architecture that supports these relationships. The two individual documents, *WS-Federation Active* and *WS-Federation Passive,* describe how to implement individual federation solutions.

  The specification is available from IBM developerWorks:

  `http://www.ibm.com/developerworks/webservices/library/ws-fed/`

- *WS-Federation: Active Requestor Profile* describes how to implement federation functionality in the active client environment. Active clients are those that are *Web services-enabled*, that is, able to issue Web services requests and react to a Web services response. Leveraging the Web services security stack, WS-Federation Active describes how to implement the advantages of a federation relationship, including single sign-on, in an active client environment.

The specification is available from IBM developerWorks:

`http://www.ibm.com/developerworks/webservices/library/ws-fedact/`

- *WS-Federation: Passive Requestor Profile* describes how to implement federation functionality in a passive client environment. A passive client is one that is not Web services enabled. The most commonly encountered example of a passive client is a *basic HTTP browser*. WS-Federation Passive describes how to leverage the advantages of a federation relationship such as single- sign-on in a passive client environment. Because this solution leverages the WS-Security foundation of the infrastructure support, the same components used to provide a passive client solution may be leveraged for an active client solution. The specification is available from IBM developerWorks:

  `http://www.ibm.com/developerworks/webservices/library/ws-fedpass/`

The logical architecture described in WS-Federation, together with the functionality described in the Web services security stack, supports both the active and passive client scenarios. The complete family of WS-Security specifications provides companies with a standards-based interoperable secure digital identity and trust platform for a Web services-based architecture. Furthermore, these specifications promote reusability of existing IT security investments, enabling companies to work with multiple security token types and multiple scenarios including basic browsers, enhanced browsers, active clients, and application-to-application connectivity.

WS-Federation allows for pull and push modes for single sign-on (SSO):

- *Pull mode* means that the SSO is initiated at the service provider. The service provider determines the identity provider, requests SSO from the identity provider, and then the identity provider responds with an SSO token.
  Refer to “Pull mode” on page 53.
- *Push mode* means that the SSO is initiated at the identity provider. The identity provider sends the SSO token to the service provider.
  Refer to “Push mode” on page 55.

## Pull mode

In Figure 2-9 on page 54, a single sign-on is triggered at the service provider by sending a special SSO trigger message to the service provider WS-Federation endpoint. If the service provider has multiple identity providers configured, it must determine which to send the client to for authentication. It can do this either by reading a cookie set on a previous visit, checking for a parameter in the query string of the SSO trigger, or by sending the user a list of identity providers to choose from.

After the service provider has determined the correct identity provider, it builds a SSO Request message, which is sent to the identity provider. The SSO message is sent in the query-string of a redirection to the WS-Federation endpoint of the identity provider. A cookie set in the redirect identifies the identity provider. It is a persistent cookie that allows the service provider to determine the correct identity provider next time, without having to prompt the user. The SSO request shown here is being sent as a result of a redirection from the service provider.

When the identity provider receives the SSO Request at its WS-Federation endpoint, it first authenticates the user (if the user is currently unauthenticated). It must have an authenticated session in order to process an SSO request. The identity provider reads the SSO request from the service provider and builds an appropriate SSO response message for that provider. This message includes a security token that is valid for the service provider.

The SSO response (including the security token) is returned to the service provider as a scripted post. The SSO message is sent to the client in the hidden inputs of an HTML form. Scripting in the form causes it to automatically be posted to the WS-Federation endpoint of the service provider. The service provider validates the received security token and uses it to build an authenticated session. It is then able to authorize the original request.

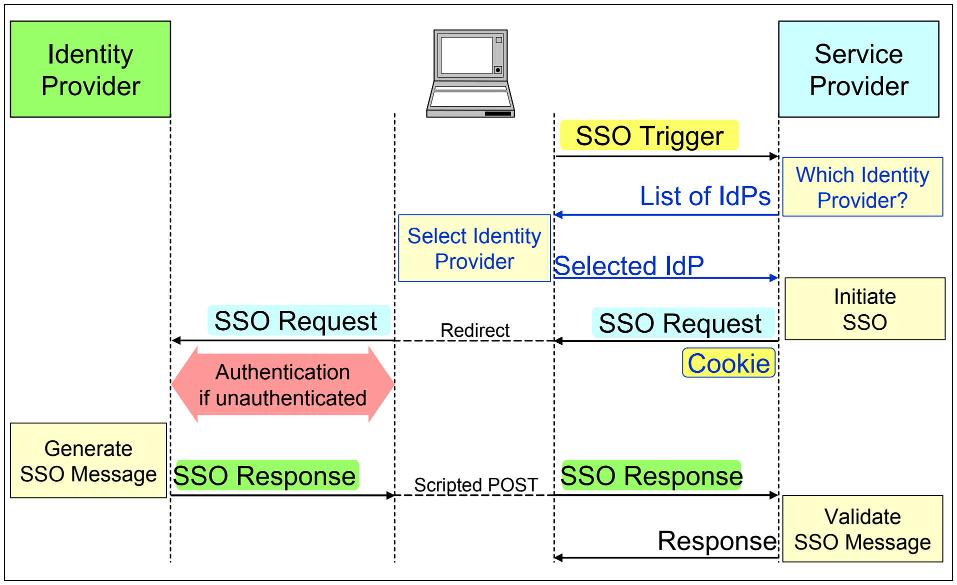


*Figure 2-9 WS-Federation: Select ID Provider and SSO (pull)*

## Push mode

Figure 2-10 shows the protocol flow for a WS-Federation PUSH operation. The WS-Federation protocol really starts with the SSO Request received from the client. However, it is useful to see what causes the SSO request to be received, so this is also included.

A user would not typically type an SSO request message directly into the browser (although the user might); a more likely case is that an identity provider will include a link on its site that a user can select to be able to access a service provider resource (in our previous example, for ABCBigCorp you would see the message `Click here to book a hotel with our preferred partner ABCRBTravel`). Rather than direct the user straight to the service provider (only for it to have to direct the user back to perform SSO), this *special* link generates an SSO request to the WS-Federation endpoint of the identity provider, which immediately triggers the SSO exchange.

This SSO request generated by the link has *exactly* the same format as the SSO request that would have been received from the service provider if it had generated the SSO message (in a *pull* operation). From here, processing is the same as for a *pull* operation. The identity provider generates the appropriate security token for the service provider and sends it to the service provider, through the client, using an HTML Form.

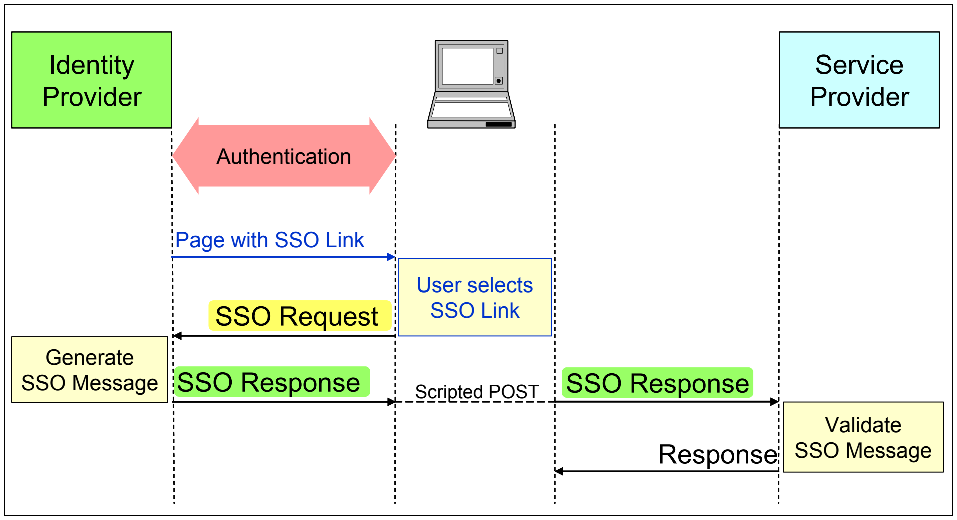


*Figure 2-10 WS-Federation: SSO (Push)*

WS-Federation also supports single logout at both the SP and IdP.

## 2.2.6 WS-Security

Although WS-Security itself is not a federation or single sign-on specification, it does define the binding of Web services security tokens. This binding is leveraged within the WS-Federation profile (see 2.2.5, "WS-Federation" on page 52).

The OASIS Security Services Technical Council, together with the OASIS Web Services Security Technical Council (WSS-TC), has defined a Web services security SAML token profile. This describes how to bind a SAML assertion *in the context of WSS:SOAP Message Security, for securing SOAP message exchanges*.

The OASIS WSS-TC issued OASIS Web services security as a specification in April 2004. Included in this specification are SOAP message security, a user name token profile, and an X.509 token profile. More information about the OASIS Web services security specification is available from:

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

## 2.2.7 Security Assertion Markup Language

Security Association Markup Language (SAML) is a standard produced by the Security Services Technical Committee (SSTC) within the OASIS Standards Organization. SAML consists of two distinct pieces of functionality: *SAML assertion* (used to transfer information about a user) and *SAML protocol* (the means of exchanging a SAML assertion). Full details are available from:

http://www.oasis-open.org/committees/security

As a protocol, SAML has three versions: SAML 1.0, 1.1, and SAML 2.0. SAML 1.0 and SAML 1.1 (collectively, SAML 1.x) focus on single sign-on functionality. SAML 2.0 represents a major functional improvement over SAML 1.x. SAML 2.0 (approved in March 2005) is based on SAML 1.x with significant input from the Liberty Alliance ID-FF and Shibboleth specifications.

SAML 1.x defines protocols for implementing single sign-on. These protocols are HTTP-redirect-based and involve the user's browser. SAML 1.x defines two profiles for these single sign-on protocols: *HTTP-based GET* and *HTTP-based POST* profiles. With the HTTP-based GET profile, the SAML assertion itself is *not* included in the HTTP-redirect response. Instead, a SAML artifact is sent from the IdP to the SP. The SP then uses an XML/HTTP back-channel to exchange this artifact for the appropriate SAML assertion.

## SAML 1.x

SAML 1.0 and 1.1 (both ratified as standards) define push-based protocols, meaning that the SSO request is initiated from the identity provider and pushed to the service provider. SAML provides for:

- Browser/POST profile
- Browser/Artifact profile

The difference between these two is how the actual security information (vouch for token) is exchanged between an identity provider and service provider.

### *Browser/POST profile*

With a Browser/POST profile, a SAML assertion (vouch-for token) is included in the response that is sent to the service provider as part of an HTML form as in Figure 2-11. This is a *front channel* exchange of the SAML assertion.

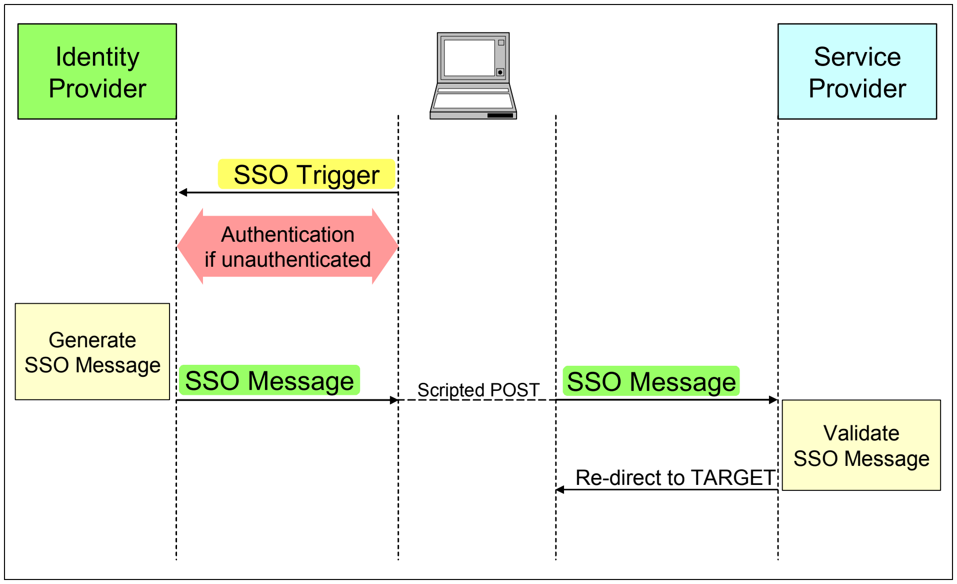


*Figure 2-11 SAML SSO: Browser POST*

### *Browser/Artifact profile*

With a Browser/Artifact profile, a pointer to the SAML assertion (called an *artifact*) is included in the query_string of an HTTP 302 redirection to the service provider. The service provider in turn issues a direct SOAP/HTTP request back to the identity provider, exchanging the artifact for the actual SAML assertion.

Both SAML profiles are invoked by a user being directed to an *Inter-Site Transfer Service* at the identity provider. The Inter-Site Transfer Service is a URL that corresponds to a FIM endpoint. See Figure 2-12 on page 58.

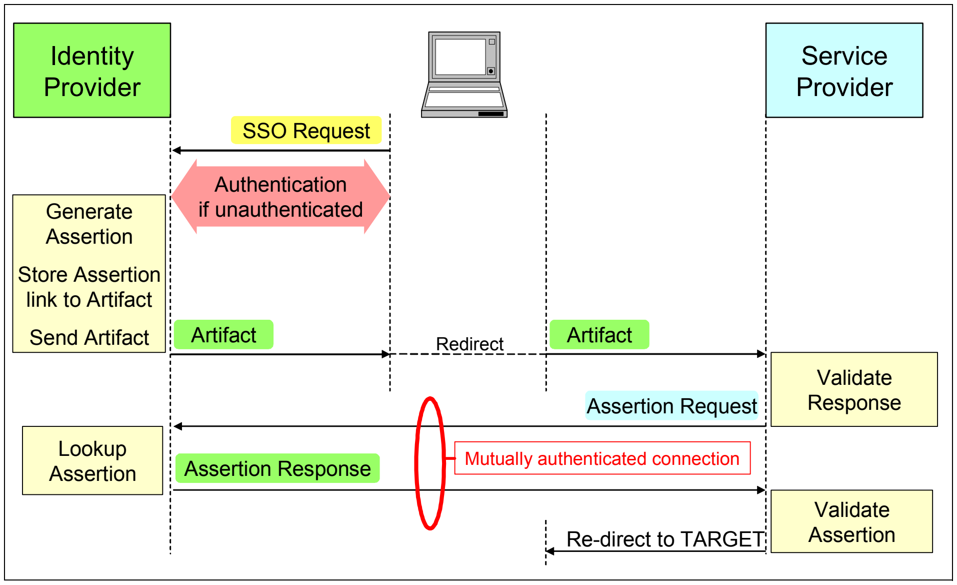


*Figure 2-12 SAML SSO: Browser/Artifact*

In Figure 2-12, the Browser/Artifact profile is shown; the step wherein a direct SOAP/HTTP request is made from the service provider to the identity provider to exchange the browser-artifact for the appropriate SAML assertion is done over the mutually authenticated connection: the back channel.

## SAML 2.0

SAML 2.0 adds single sign-out and account linking functionality in addition to *enhanced client/proxy* support in aid of mobile device support. As part of this increased functionary, SAML 2.0 provides a richer set of profile bindings, including artifact and assertion versions of all of the requests/responses leveraged over HTTP-based GET and HTTP-based POST profiles.

As the most recent release, SAML 2.0 has been extended with input from both the Shibboleth work and Liberty ID-FF 1.2. SAML 2.0 takes into account more of the identity life cycle functionality than previous versions. Likewise, based on the Shibboleth input, SAML 2.0 has functionality that addresses some of the privacy concerns associated with a federated environment.

## Liberty

The *Liberty Alliance Project* was formed to deliver and support a federated network identity solution for the Internet that enables single sign-on for consumers and business users in an open, federated way.

The Liberty Alliance Identity Federation Framework (ID-FF) extends SAML functionality beyond the push-based single sign-on of SAML. Tivoli Federated Identity Manager SPS supports Liberty 1.1 and 1.2 ID-FF.

Tivoli Federated Identity Manager trust service supports Liberty Assertions. ID-FF defines:

- Pull-based SSO protocols
- Functionality for single logout (SLO)
- Account linking and de-linking:
  - Liberty Register Name Identifier (RNI) profile
  - Liberty Federation Termination Notification (FTN) profile
- *Where are you from?* (WAYF)
  - Liberty Identity Provider Introduction (IPI) profile
- Unsolicited authentication response
  - Allows a push SSO to take place; SSO initiated by the identity provider.

The ID-FF single sign-on protocols have three flavors:

- Browser/Artifact (B/A)
- Browser/POST (B/P)
- Liberty-Enabled Client/Proxy (LECP)

For more information about Liberty Alliance, and for details about ID-FF single sign-on protocols, go to:

`http://www.projectliberty.org`

For information about Federated Identity Management, see the Business Forum Web site:

`http://www.bizforum.org/whitepapers/ibm-4.htm`

### Liberty Register Name Identifier

The Liberty Register Name Identifier (RNI) profile is used to manage a user's pseudonym (NameIdentifier). The Liberty NameIdentifier is used for account linking purposes. In a Liberty environment, the establishment of such a pseudonym is part of the process of federation; without this process, a single sign-on protocol cannot be completed.

The Liberty *NameIdentity* is set during a specialized single sign-on request, a *federation* request. Subsequent *NameIdentifier* management processing may be initiated by an identity provider or a service provider.

In general, an identity or service provider may automatically reset the name identifier values on a periodic basis (as defined within the relationship) in response to an end-user-initiated request, or in response to some administrator trigger. An example of an administrator trigger at an identity provider would be a request to set new (identity provider-provided) name identifiers for all users federated with a particular service provider.

### Liberty Federation Termination Notification

The Liberty Federation Termination Notification (FTN) profile defines the process by which an account linking is removed. This is also referred to as de-federation. De-federation removes the account linking maintained by a NameIdentifier.

In general, an identity or service provider will initiate a FTN request in response to an end-user-initiated request or in response to some administrator trigger. An example of an administrator trigger at an identity provider would be a request to terminate the account linking information for all users federated with a particular service provider (perhaps in response to a high-level termination of the overall business relationship).

### LIberty Single Logout

The Liberty Single Logout (SLO) profile defines the process by which a valid session (set of valid sessions) for a user is destroyed. Single logout can be initiated in response to a user request at an identity provider or a service provider with whom he has a currently valid session. A SLO request received at a service provider will in turn cause an SLO action at the identify provider, where the IdP in turn logs the user out of all currently valid SP sessions *except* the SP session that initiated the IdP logout.

Note that although logout is almost always a user-initiated process, there may be situations in which either business partner must immediately terminate all sessions and thus issue a logout request on behalf of the user. This process can occur, for example, within a business environment in which an employee is fired for misconduct; all currently valid sessions for the user must be terminated as the employee is escorted off the employer's premises. In this case, the SOAP SLO profile can be leveraged, because it might occur out-of-band (without waiting for a user interaction at either side).

### Identity provider introduction

The Liberty identity provider introduction (IPI) profile defines the process by which an identity provider can set, and a service provider retrieve, a *common domain cookie* (CDC). This cookie is defined for a common domain, a DNS alias shared by identity business partners and service providers within a *circle of trust*. It is used to store information that is accessible or required by all business partners within the circle of trust, in particular, the user’s identity provider. After retrieved, the information contained in the cookie is extracted and returned to the requested domain using techniques such as URL rewriting.

### *Liberty-enabled client/proxy*

The Liberty-enabled client/proxy (LECP) profile is designed to address devices that are not able to accommodate the query-string length requirements of the Browser/Artifact (B/A) profile or the form post requirements of the Browser/POST (B/P) profile. These devices are generally mobile devices, such as query-string length limited mobile devices or older mobile devices that are not capable of automating a form post.

A Liberty-enabled client is a client that has, or knows how to obtain, knowledge about the identity provider that the principal wants to use with the service provider. This process may be implemented as a client (for example, code downloaded to a mobile handset) or as a proxy (for example, an HTTP proxy embedded in a WAP gateway). In addition, a Liberty-enabled client receives and sends Liberty messages in the body of HTTP requests and responses. Therefore, Liberty-enabled clients have no restrictions on the size of the Liberty protocol messages.

Figure 2-13 shows the role of Tivoli Federated Identity Manager in an LECP profile, where a WAP Gateway is acting as the LECP. In this scenario, Tivoli Federated Identity Manager accommodates only steps 4 and 6 when acting as an identity provider, and steps 1, 3, 7, and 11 when acting as a service provider.

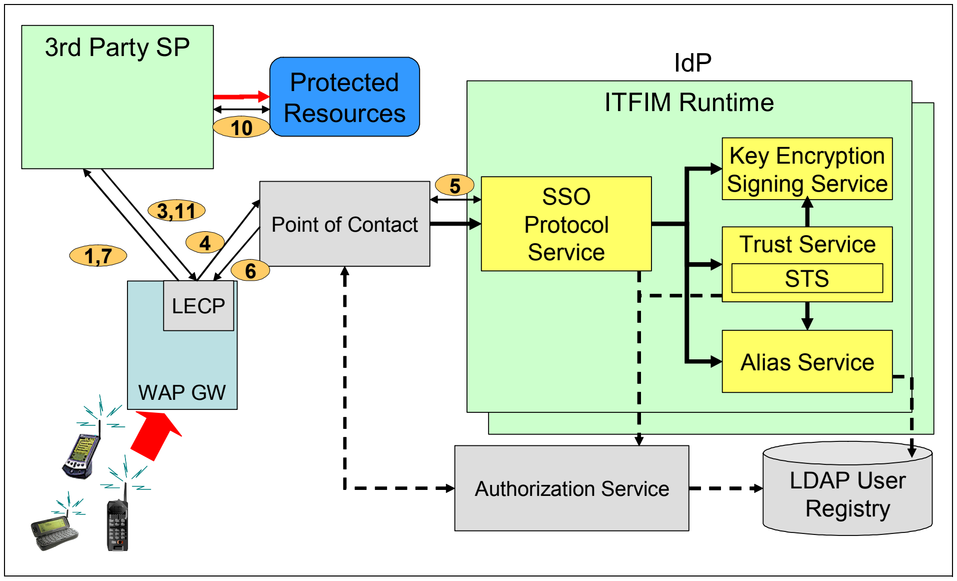


*Figure 2-13 Liberty enabled client proxy (LECP) example*

Details about the LECP profile are given in the following Liberty Alliance specifications, listed here with links to the PDF files:

- [*liberty-architecture-bindings-profiles-v1.1*]

  `http://www.projectliberty.org/liberty/content/download/1469/9656/file/liberty-architecture-bindings-profiles-v1.1.pdf`

- [*liberty-architecture-protocols-schema-v1.1*]

  `http://www.projectliberty.org/liberty/content/download/1468/9653/file/liberty-architecture-protocols-schema-v1.1.pdf`

### *Unsolicited authentication response*

Liberty 1.2 allows for an identity provider to send an *unsolicited authentication response* to a service provider, allowing a push SSO to occur (an SSO initiated by the identity provider). The trigger for this is not specified, so the person who implements is the one who decides.

## 2.3 Federated single sign-on

*Federated single sign-on* (F-SSO) is the process by which a Web-based user authenticates to a federation business partner, identity provider (IdP), and has the IdP assert a relevant identity (and attributes) to any or all required service providers (SP) as part of the user's online federation experience.

Global sign-on itself is provided by a federated single-sign-on protocol that provides standard, interoperable means for multiple federation business partners to negotiate the presentation of credentials about a user from an identity provider to a trusted federation service provider. These protocols are explained in more detail further in this chapter.

When considering a single sign-on solution, the two main areas where participants must agree on the technology choice in order to achieve interoperability are.

- The format and content of the security token that will be passed between the partners

  The security token generated by the sending partner must be understandable by the receiving partner. Also, there must be an agreement as to what information is sent in the token and how it is interpreted. Typically, the security token format is bound to the single sign-on protocol (SAML protocols use SAML assertions; Liberty ID-FF protocols use Liberty specializations of SAML assertions). With Tivoli Federated Identity Manager, security token generation and consumption is handled by the trust service as invoked internally by the single sign-on protocol service. This is discussed in more detail in “Trust services” on page 74.

- The single sign-on protocol

  This area defines how the parties will communicate. A single sign-on server must know how a client will request a security token and how the token should be packaged and returned. The server must also know how a client will present an incoming security token in order to initiate an authenticated session. In Tivoli Federated Identity Manager, all single sign-on protocol messages are handled by the single sign-on protocol service.

Note that a single sign-on standard does not only deal with a profile for single sign-on, but also profiles for single logout, federation, and alias management. The SSO Protocol Service is also responsible for handling these messages.

Figure 2-14 shows the communications and exchanges that take place at each layer of Tivoli Federated Identity Manager when performing Web-based single sign-on. Note that no internal details are shown for the third-party side because their architecture is not known (and not important).

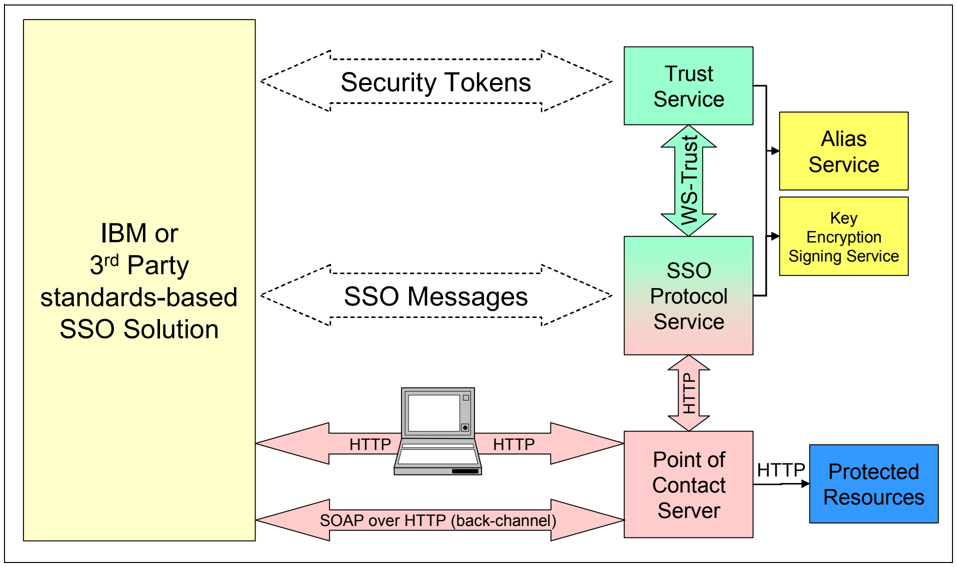


*Figure 2-14 Single sign-on components and communication*

At the *communication* layer, HTTP messages are being handled by the point-of-contact server. In the IBM solution, this is WebSEAL. All real communication is through the point-of-contact server. It must support the HTTP standard in order to interoperate with the client and with the third-party solution.

At the *protocol* layer, SSO messages are being exchanged between Tivoli Federated Identity Manager and the third-party solution. In Tivoli Federated Identity Manager, this layer is handled by the SSO Protocol Service. It exchanges SSO messages with the third-party solution through the point-of-contact server.

At the *trust* layer, security tokens are being exchanged between Tivoli Federated Identity Manager and the third-party solution. In Tivoli Federated Identity Manager, this layer is handled by the trust service. The trust service exchanges security tokens with the third-party solution through the SSO Protocol Service.

## 2.3.1 Architecture for F-SSO

Figure 2-15 shows the Tivoli Federated Identity Manager architecture required to support Web-based single sign-on protocols such as Liberty, WS-Federation, and SAML.

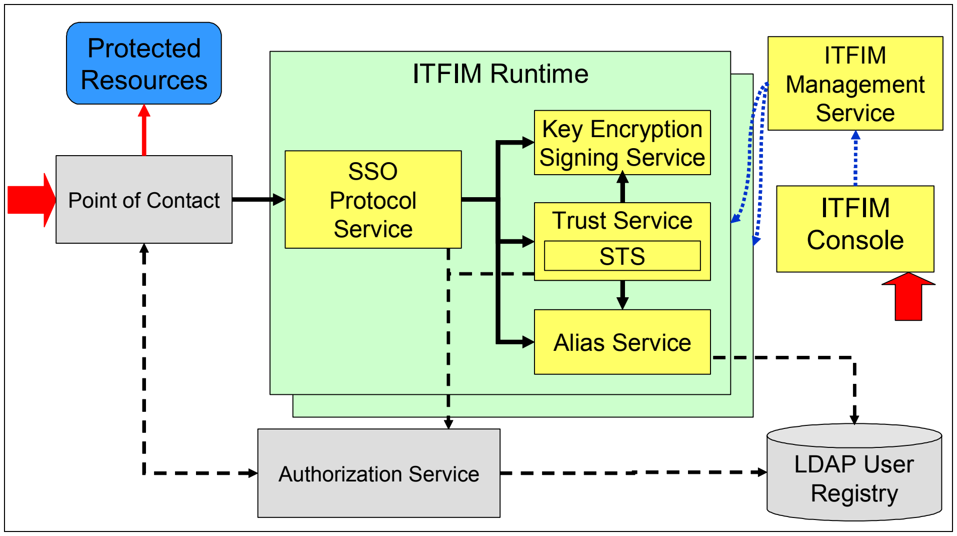


*Figure 2-15 Tivoli Federated Identity Manager components for Federated SSO*

All outside communication with the environment comes through the HTTP point-of-contact server, in this case WebSEAL. WebSEAL maintains the Web session with the client and manages authorization. WebSEAL also triggers authentication (either local or SSO) when non-public resources are requested. WebSEAL authorization is managed by authorization service, in this case the Tivoli Access Manager.

WebSEAL has a junction to the SSO Protocol Service (SPS). Incoming SSO messages will be directed to the junction that connects to the SPS. WebSEAL will simply forward these as normal. WebSEAL can also re-direct the client to the SPS in order to initiate SSO processes itself.

The SPS communicates with the trust infrastructure components in order to build and consume SSO messages and uses the Access Manager Administration APIs in the authorization service to terminate WebSEAL sessions during single logout (SLO) operations.

The Tivoli Federated Identity Manager environment is managed using the Tivoli Federated Identity Manager Console. When a federation that includes SSO functionality is configured, the Tivoli Federated Identity Manager console updates the SPS configuration as appropriate to support this.

## 2.3.2 Trust in F-SSO

Security tokens are included in a message to pass an identity and to convey security-specific information (used for authentication or authorization purposes, for example) about a requestor; see Figure 2-16. These tokens are common to (at least) one other business partner and contain pre-arranged security-relevant information. These tokens are themselves protected through signing and encryption, often using the same keying material as used at the message layer. This information is part of the trust infrastructure in the same way that keys are used for signing and encryption purposes. The proper use of these tokens conveys information about the holder of these tokens. The trust service provides a means of managing these security tokens and the trust relationships bound to these security tokens.

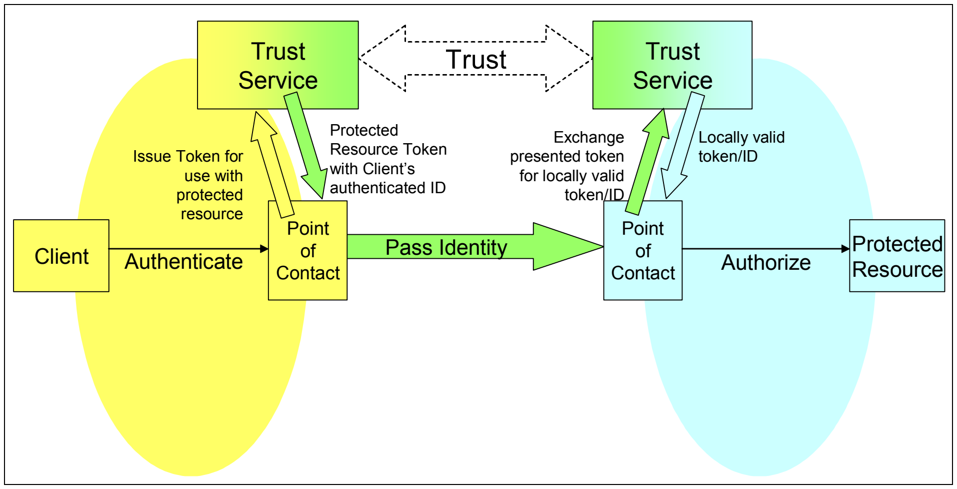

Trust Service
Trust
Trust Service
Issue Token for use with protected resource
Protected Resource Token with Client's authenticated ID
Exchange presented token for locally valid token/ID
Locally valid token/ID
Client
Authenticate
Point of Contact
Pass Identity
Point of Contact
Authorize
Protected Resource

*Figure 2-16 Using trust service in F-SSO*

Token management is based on information such as the issuer of a token, the intended destination of the token, and the intended use of the token. This approach allows the trust service to manage a business partner's token meta-data together with the business partner's cryptographic material.

## 2.3.3 F-SSO protocol functionality

Tivoli Federated Identity Manager and Access Manager for e-business together provide support for browser-based federated single sign-on (F-SSO) protocols. F-SSO protocols differ from earlier attempts at cross-domain single-sign-on protocols in their enhanced functionality, such as single sign-off. In this section we briefly describe the type of functionality found in single sign-on and F-SSO protocols.

### Single sign-on

Single sign-on (SSO) is a widely-understood process. It is the process of allowing a user, authenticated to one domain (the user's *home domain* in Access Manager terms, also known as the user's identity provider) to present an assertion or token (a *vouch-for* token in Access Manager terms) to a business partner (also known as a service provider) as proof of authentication. This token is used to identify the user and build a locally valid session (including credentials) for the user without having to prompt the user for authentication credentials.

In general, F-SSO protocols (and all other cross-domain SSO protocols) are of two types: *push* and *pull*.

- Push protocol

  In push protocol, the user invokes a remote resource from within the control of the user's home domain (through a link on a portal page, for example), and is redirected to the remote resource, carrying the user's vouch-for token with the request. This means that the service provider (site of the remote resource) does not have to prompt the user for information about the user's home domain or prompt the user's home domain for vouch-for information. Push protocols are limited in that they must be invoked from within the control of the user's home domain, and push protocol scenarios do not handle bookmarked URLs or direct-typed URLs.

- Pull protocol

  In pull protocol, a user invokes a remote resource at a site other than their home domain (the service provider domain). Because the service provider is not able to authenticate the user, the service provider must determine the user's home domain and then request SSO information from the user's home domain.

The process of determining the user's home domain is often referred to as *WAYF*, or *Where Are You From*. WAYF may be established based on a long-term set of information carried around by the user (for example, in the form of a domain cookie identifying the user's home domain) or by an explicit user interaction, where the user is prompted to identify the home domain (for example, from a pre-configured list of service provider-trusted home domains). Pull protocols are limited in that if the service provider is not able to determine the user's identity provider without user interaction, then a user-driven WAYF sequence is required (for example, on first access to a service provider or after a cookie-cache-flush).

After a user's single sign-on information has been established and validated at a service provider, the service provider will maintain a local session (including credentials) for the user. This approach allows the service provider to implement local access control policies, for example, for the user's session.

## Single logout

Previous attempts at single sign-on have often neglected the corresponding single logout (SLO) functionality. Logout can be of two forms: Local and global. In general, logout from the user's identity provider should force a global logout, whether the user requests a global or local logout. This requirement is the result, in large part, from the liability normally assumed by an identity provider for a user within an F-SSO relationship.

Logout does not always have to be an allowable service provider action. If a user has performed a single signed-on to the service provider, the user might not realize that this is a separate session with this service provider. Rather than confuse the user by offering a logout action at the service provider, we expect that most scenarios will set a short-session lifetime (inactivity timeout) at a service provider and rely on SSO to re-establish a session at a service provider, perhaps many times within the lifetime of the user's identity provider session.

Presenting a user with a global logout option at the service provider should trigger a logout notification to the user's identity provider and then a logout attempt from the service provider. The global logout received at the identity provider should then invoke global logout functionality by the identity provider, followed by local logout at the identity provider.

**Note:** Logout can have implications for things such as session duration (differing durations at identity providers and service providers). In general, the inacitvity timeout that is set for an identity provider should be longer than that set for its service provider business partners. This prevents a user from timing out at the identity provider when executing a lengthy transaction with a given service provider.

## Account linking

Account linking is the process of the runtime linking of a user's accounts at different business partners. Accounts are linked by establishing some form of *common unique identifier* that is shared by different business partners, and locally mapped at the business partner site to the user's local identity. This common unique identifier is usually defined to contain no information about the user, so that it cannot be easily reproduced by outside parties (including malicious third parties). As such, this common unique identifier is often referred to as an *alias* or a *pseudonym*. Account linking is also known as *name federation* within Liberty Alliance specifications.

Account linking is a required functionality when a user wants participation in a federation but already has existing accounts at both federation business partners (assuming a federation of two). In order for single sign-on to succeed, the identity provider and service provider must have a common way of identifying the user. Account linkage is the process of establishing this linkage, based on an initial user interaction at both the identity provider and service provider side. This means that as part of the account linking process, a write operation to an identity store will occur to allow the saving of the linking/mapping information.

In some cases, the account linking process sets a user's authentication information, at the service provider, to a disabled state. This means that as a result of the federation, the service provider can no longer directly authenticate the user but will always refer to the linked identity provider for this information. The service provider may choose to keep the user's pre-account linking password so that when a user de-federates the accounts, that user may still access the service provider information based on direct authentication to the service provider (or single sign-on from a new, different identity provider).

Note that account linking is sometimes referred to as provisioning, where the linkage between existing accounts is the information being provisioned. This, however, is *not* provisioning for two important reasons:

- It requires that the user already has pre-existing accounts at both the identity and service provider.
- The account linking requires that a user be actively involved in the process of establishing the account linking at both providers.

Tivoli Federated Identity Manager does provide a Web services provisioning solution. This Web services-based provisioning allows the linking of two Identity Management systems for a complete user life cycle management solution, including the provisioning of information (attributes, subscriptions, account status, and so on) between federation business partners.

## 2.4 Federation services

Tivoli Federated Identity Manager services facilitates a standardized means for allowing businesses to perform the following tasks (see Figure 2-17 on page 70):

- Engage in trust relationships that facilitate direct integration of business processes in the most efficient fashion. The concept of business federations directly provides services for customers registered at other (business partner) businesses or institutions by establishing business trust relationships.
- Share identity information and entitlements in a trusted fashion between companies. Current approaches to identity management generally rely on companies incurring user life cycle management costs by maintaining redundant identities to manage employees, business partners, and customers. The relationship between the business and these individuals can change fairly frequently. Each change requires an administrative action that can result in a high cost of user life cycle management.
- Exchange, in a secure and trusted manner, tokens that refer to a principal, the principal's attributes, privileges, and so on. These tokens are used to communicate information used for the authentication and authorization of a principal to a business partner.
- Maintain security in a Web services-oriented architecture, allowing for secure standards-based application-to-application inter-enterprise communication.

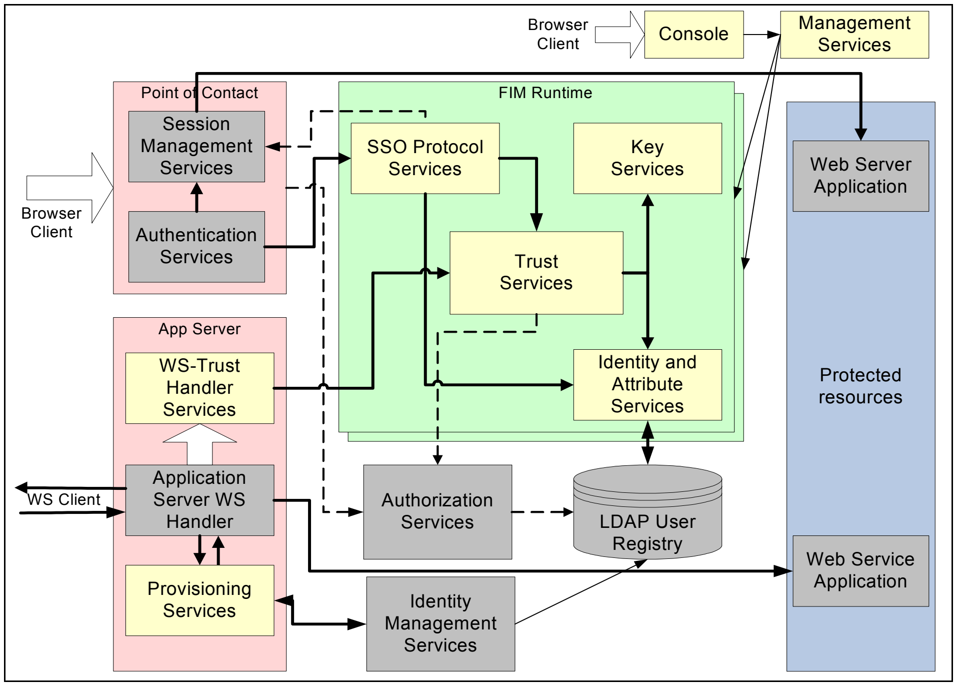


*Figure 2-17 FIM services architecture - The full picture*

The following sections give an overview of each of the services components represented in Figure 2-17, which is the FIM services architecture used in Tivoli Federated Identity Manager. The complete set of Tivoli Federated Identity Manager services allows for creation of federated SSO, Web services security management, and provisioning solutions. The dark-gray boxes in the figure are non-core Tivoli Federated Identity Manager services that are used as part of different FIM solutions.

A different view of the services is found in Figure 2-18 on page 71, where the layers PoC, SSO protocol service (SPS), and trust service are shown in their external communication interfaces over standardized protocols. Both user-based and application-based interactions are shown, because they differ in layering and protocols.

In the application-based interaction to the left in Figure 2-18 on page 71 the PoC is represented by a WS handler interfacing with the trust service, and may be represented by a WS firewall/gateway. The provisioning service may be viewed as an application exposed as a Web service.

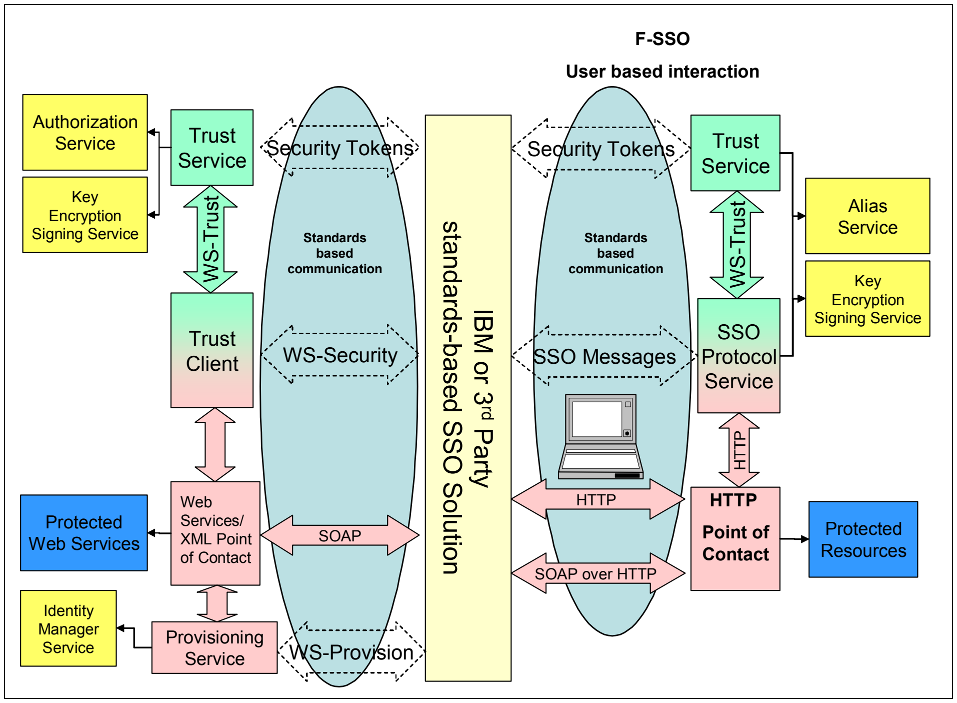


*Figure 2-18 User and application-based interaction components and communication*

## 2.4.1 Point of contact

The *point of contact* (PoC) is used for HTTP-based user interactions. The PoC service provides authentication service and the session management service functionality. These services are typically provided by Tivoli Access Manager for e-business through the Access Manager for e-business reverse proxy or the Access Manager for e-business Web plug-in.

### Authentication services

Authentication services provide the functionality required to evaluate and validate user-provided credentials. Authentication services evaluate credentials such as a user name and password, secure ID token pass phrases, X.509 certifications, and so on, directly provided by a user. Authentication services are able to invoke some backend data stores, such as a LDAP registry or a secure ID token server, to validate these credentials.

The protocol used to collect authentication credentials from a user requires a simple challenge/response interaction with the user. The process of evaluating these credentials is typically a simple action such as an LDAP-based validation of presented credentials. After the successful validation of authentication credentials, the authentication service presents the session management service with the information required to build a session for a user.

In a simple direct user authentication environment, a challenge/response protocol to collect the user's authentication credentials is negotiated directly between the user (or a user agent such as the browser) and the authentication service.

Within a F-SSO environment, the challenge/response protocol is not always negotiated with the user but may be negotiated with a third party acting on behalf of the user. This third party usually asserts a form of security token or assertion about the user, based in its own (local) authentication of the user. This security token acts as the equivalent of the user-presented credentials. This security token must be validated, but this validation is based on the trust relationship between the business partners.

Instead of incorporating support for each of these federation protocols (both the interaction with the business partner and the evaluation of the presented token) within the authentication service, an external SSO service is used. SSO services (described in 2.4.2, "Single sign-on protocol services" on page 73) provide the run time for the federation protocols necessary to implement the challenge/response interaction with a third party.

In response to the evaluation of user-provided or federation-provided authentication credentials, an authentication service generates information that is used by a session management service to govern a user's session. This information is typically represented as a set of user credentials, or user privileges. This information is used by a session management service and by *authorization services*, as described in 2.4.5, "Authorization services" on page 79.

### Session management services

The purpose of a *session management service* is to manage a user's session life cycle, from session creation, to session access, to session deletion (in response to session logout services). These services typically sit at the edge of a network, where they guide a user's access requests and access experience within an enterprise.

Sessions are created at a session management service in response to a successful authentication or a successful security token validation. Current implementations of session management services often incorporate *authentication services*, so that an authentication service exists as a logical service.

## 2.4.2 Single sign-on protocol services

Within a federation environment, federated identity management protocols are used to communicate information about a user between federation business partners. For example, with F-SSO the result of this communication is some form of security token that must be validated; this token provides the information required to determine a user's local identity. Federation single sign-on protocols provide a vendor-neutral means of establishing the communications required to exchange this security token.

In Tivoli Federated Identity Manager, the responsibility for handling single sign-on protocol messages is off-loaded from the point-of-contact server, as shown in Figure 2-19. Single sign-on protocol endpoints are instead hosted by a separate service, the single sign-on protocol service (SPS). The point-of-contact server still maintains control of user sessions, providing session management services.

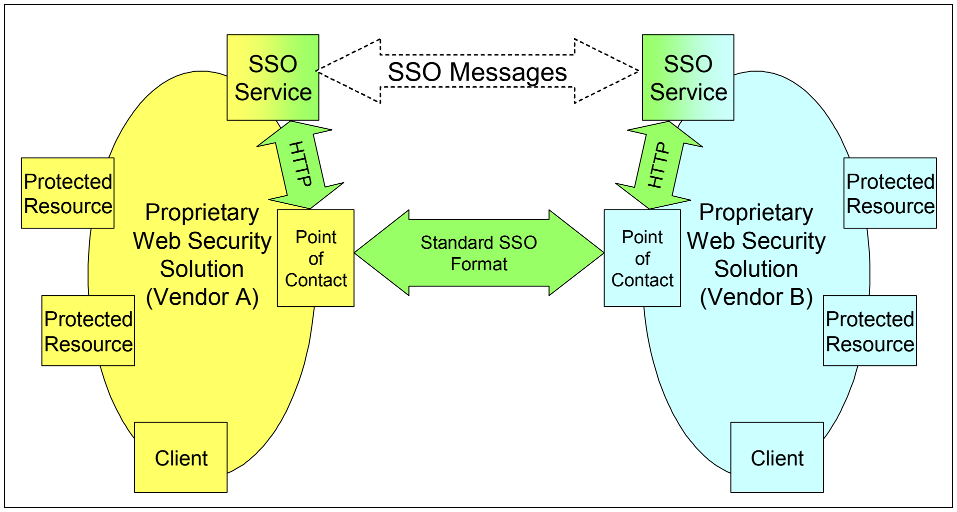


*Figure 2-19 Externalized SSO services*

The point-of-contact server has a number of interfaces to the SPS but these do not have to be modified to be able to support different (or new) SSO standards. Only the SPS has to be modified if changes to SSO behavior are necessary.

### External authentication interface

Tivoli Federated Identity Manager provides an authentication mechanism through its SPS with the capability that allows clients to sign in with credentials generated by another party (the identity provider). By integrating Tivoli Federated Identity Manager with a point of contact, the federated SSO can be treated as just another point-of-contact authentication mechanism, thus having the SPS create an point-of-contact login session. When used with Tivoli Access Manager as the point-of-contact service, the external authentication interface (EAI) is used as the integration point with Tivoli Federated Identity Manager.

## Trust services

Federation relationships require a trust relationship-based federation between business partners. A trust relationship is represented by the combination of the security tokens used to exchange information about a user, the cryptographic information used to protect these security tokens (and the communications used to broker token exchange), and optionally the identity mapping rules applied to the information contained within this token.

The trust service provides the management of this overall trust relationship, including the binding of a trust relationship to a particular partner. As part of this trust relationship management, the trust service provides a means of managing one's own keys and certificates (through a key service), and of binding a business partner's certificates (validated by a third-party certificate authority) to the local, business-agreement validated, business partner identity. These keys and certificates are then used to sign/validate and encrypt/decrypt messages between business partners, independent of any transport layer security. These services provide the trust infrastructure over which other federation services are layered.

Trust services require more than just the management of cryptographic elements. This is because trust relationships are also bound to security tokens exchanged between business partners. Security tokens are managed by a *security token service* (STS). Within Tivoli Federated Identity Manager, the STS is implemented as a logical service contained within the trust management service. We call out the notion of a security token service as a separate service to highlight the difference in management required for cryptographic elements and security tokens. Below is the trust service studied in more detail.

Figure 2-20 on page 75 shows the logical components and connections of the Tivoli Federated Identity Manager trust service. The trust service performs security token related functions such as token creation, validation, and exchange, and it does authorization for Web services. The trust service is accessed by trust clients using either SOAP requests or direct JAVA API calls.

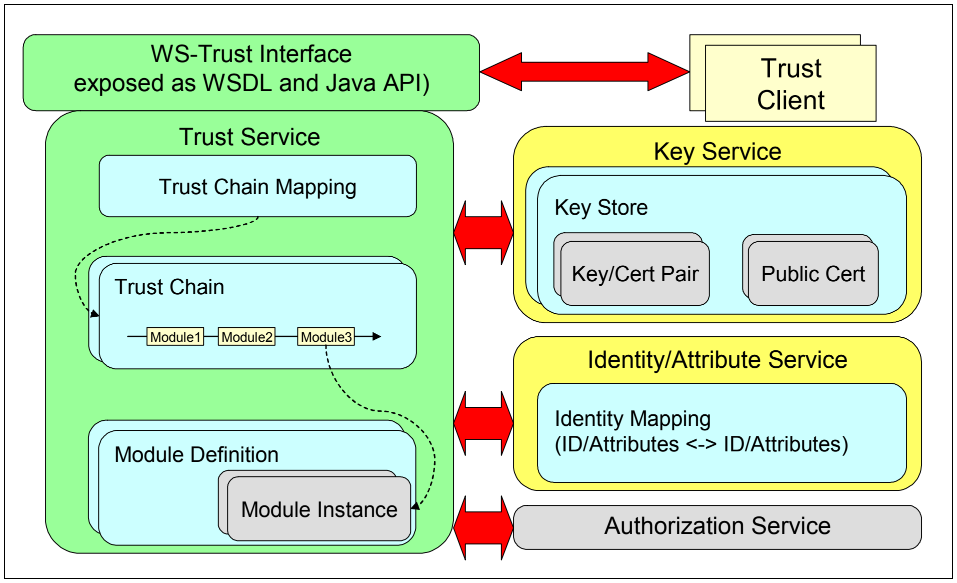


*Figure 2-20 Trust service components and connections*

## Trust service modules

All trust service functionality is performed by chains of modules. There are modules that can process incoming tokens, modules that create tokens, modules that perform identity mapping, and modules that perform authorization. A module definition points to the implementation of a module and a module instance contains the specific configuration.

Trust service modules can make calls out to other Tivoli Federated Identity Manager components. For example, most token modules call the Key Service for signature creation and validation. Liberty token modules call out to the Identity Service for alias lookup. AM Credential modules and authorization modules call out to authorization service.

When exchanging security tokens with partners, simply understanding the different token standards is not enough. Also important to know is what information a particular partner is expecting in tokens from your site, and what information you should expect to receive from partners.

For example, two different partners in the same federation might format a user account number in two different ways, and might use a different attribute in the security token to exchange it. Both partners use the same token standard for example SAML 1.1, but the information within the token is different.

The Tivoli Federated Identity Manager trust service has a very flexible identity mapping function that allows it to exchange tokens using a different identity mapping rule with each partner. The trust service mapping module is called to perform the mapping, and it looks up the configured identity mapping for the partner in question.

Information from the incoming token can be manipulated and mapped into the outgoing token in any way required. In addition, hard-coded information can be added to the outgoing token. Also possible is to use JavaScript or Java to acquire information from external sources. This flexibility is achieved by using XSL transformations for identity mapping. XSL is a very powerful transformation language and the trust service mapping module takes full advantage of its capabilities.

The trust service defines an abstract format for identity information. This format is an XML document called the STS Universal User. The two reasons for having this abstract format are:

- To allow conversion from any supported token type to any other type. The most scalable way to do this is to have each token module be able to convert from its native token type into the abstract type, and to be able to convert from the abstract type into its native token type. Then, converting from any token to any other token is possible through the abstract format.
- To be able to perform identity mapping. This mapping is made much simpler if the mapping module has to deal with only one abstract identity format, rather than multiple real identity formats. Leveraging an XML formatted STS Universal User allows us to leverage techniques such as XSLT and the many XML editors and XSLT tools for the management of this functionality.

The STS Universal User is an XML document that contains identity information in a generic way. It contains three sections: one for principal information, one for group information, and one for attribute information. In a standard SSO trust chain, an incoming token is converted to this format, the identity mapping is performed, and then the outgoing token is created.

Figure 2-21 on page 77 shows how the trust service performs a token exchange. Trust chains like the one shown here are used for all Federated SSO operations. These trust chains are created automatically when you configure Federated SSO.

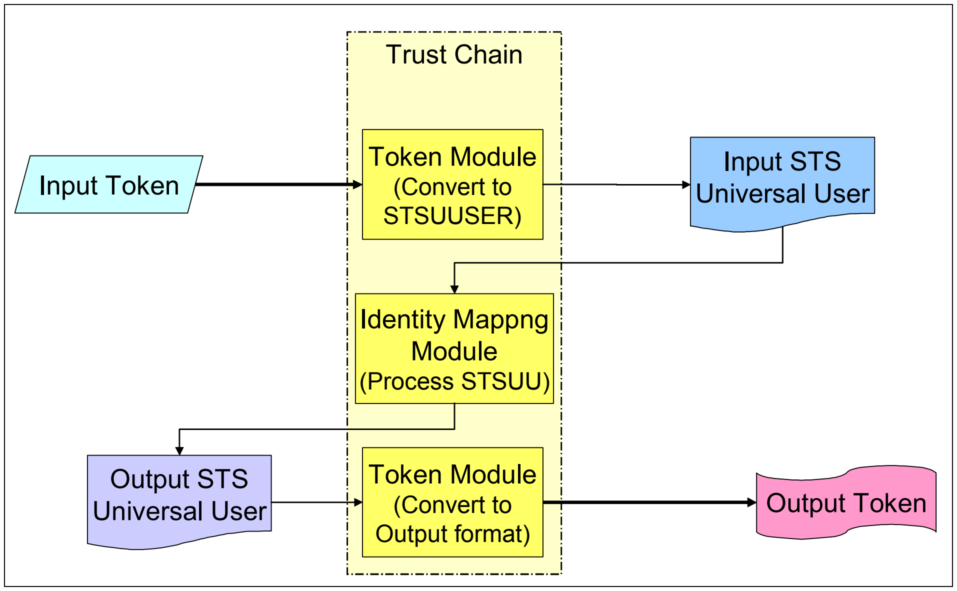

Trust Chain
Input Token
Token Module (Convert to STSUUSER)
Input STS Universal User
Identity Mappng Module (Process STSUU)
Output STS Universal User
Token Module (Convert to Output format)
Output Token

*Figure 2-21 Trust service processing for F-SSO*

The input to the trust chain is the input security token. The first module in the trust chain converts the input token to a STS Universal User (STSUUSER). This approach creates an XML document with known structure. All attributes from the incoming token are available in the STSUUSER document.

The STSUUSER document is now used as input to the identity mapping module. The mapping used by the module is particular to the partner we are dealing with and so is tailored to the particular attributes and information formats used by that partner. The output of the mapping module is another STSUUSER document, one that is suitable for creation of the outgoing token (or another mapping module or other trust chain module). The output STSUUSER document can now be converted into the output token format by the final token module.

Figure 2-22 shows how the Identity Mapping module is implemented using an XSL parser.

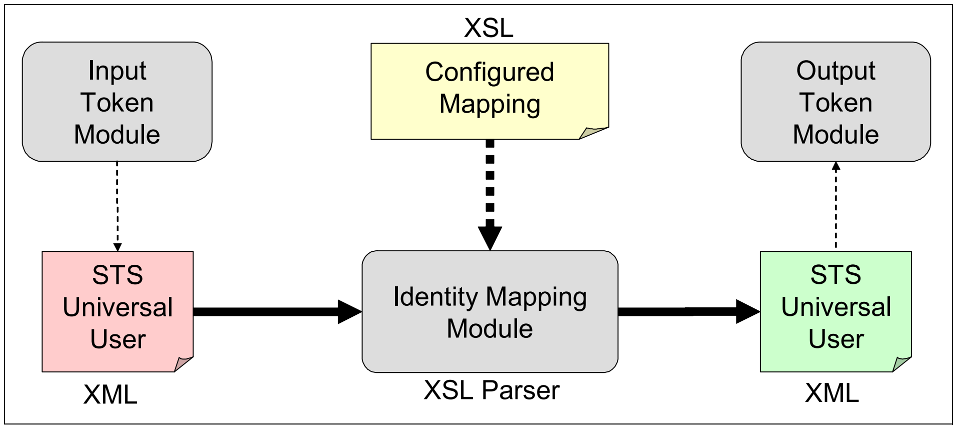


*Figure 2-22 Trust service transformation engine*

The input STSUUSER document is generated by the input token module. The input token module handles the token validation process and is responsible for correctly extracting information from the input token and building the contents of the STSUU. This STSUU is fed into the XSL parser along with the configured XSL mapping rule for the transformation.

The output of the XSL parser is another XML document. In fact, the XSL mapping rule must be such that the output document is another STSUUSER document. This STSUUSER document is fed into the output token module in order to create the required output token.

As mentioned previously, the information in the input STSUUSER document, and the information required in the STSUUSER document, is dependent on the token modules in use. The configured mapping must consider both of these things.

### 2.4.3 Key services

Key services (KESS) are leveraged to provide access to key stores used by a trust service and the SSO protocol service (SPS), allowing the trust service and SPS to plug in and access different key stores as required. It also provides a single point through which key management may be accomplished. Key services are often implemented as logical components within a trust service.

### 2.4.4 Identity services

Identity services is a generic term for those services that provide the interface to local data stores, including user registries and databases, for identity-related information management. Typically, an identity service is able to add, delete, and look up information against a backing data store.

Identity services are leveraged by many different services within a federation environment. The authentication service leverages identity service functionality as part of the evaluation of user-presented authentication credentials and to build the privilege credentials used by the session management service. These privileges are based on the attributes of a user stored within a data store (these attributes includes information such as group membership, roles, personal attributes such as age, and so on).

WIthin a Tivoli Federated Identity Manager environment, identity service functionality is leveraged as part of the identity management functionality within the trust service. This refinement of an identity service, namely an *Identity Attribute Service (IdAS)*, provides the functionality required to manage the attributes required for a security token.

An IdAS normally accesses an enterprise directory or other shared repository; this will allow the attribute services to leverage existing attribute stores and attribute management techniques.

#### Alias services

A specialized form of identity service is an *alias service*. Alias services are part of single sign-on service functionality; they are used to provide the mapping between an alias and a local user identity. Aliases are often included in the security tokens exchanged within a single sign-on protocol. They are a provider-neutral means of referring to a user. An alias service may leverage an external data store, such as an enterprise directory, for the storage of SSO aliases, or it might leverage a private, internal data store.

### 2.4.5 Authorization services

Authorization services are responsible for providing access decision point functionality within a security model. The authorization service itself might not act as an *access enforcement function* (AEF). AEF functionality is typically provided by session management services. Tivoli Access Manager provides AEF functionality with Tivoli Access Manager for e-business WebSEAL.

At their simplest, authorization services implement an access decision functionality, taking in a request for access and evaluating this request based on a user's session privileges. The authorization service might respond with a simple yes or no, indicating whether an access request is allowed. Based on this response, session management services act as the authorization enforcement point by allowing or disallowing the actual request for access.

## 2.4.6 Web services security

Web services security (WS-Security) defines a standard set of SOAP extensions that can be used when building secure Web services to implement integrity and confidentiality. This approach allows for sending security tokens to authenticate requests and signing data to ensure data integrity and verify sender. To ensure privacy of data, the data is encrypted. All this with the goal to accomplish end-to-end message content security.

The SOAP message security specification is called “Web Services Security: SOAP Message Security 1.0” and it can be found at:

http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

This standard defines a set of SOAP extensions, seen in Figure 2-23, that provide the ability to:

- Send security tokens as part of a message
- Include an XML Digital Signature as part of a message
- Encrypt all or part of the message using XML Encryption

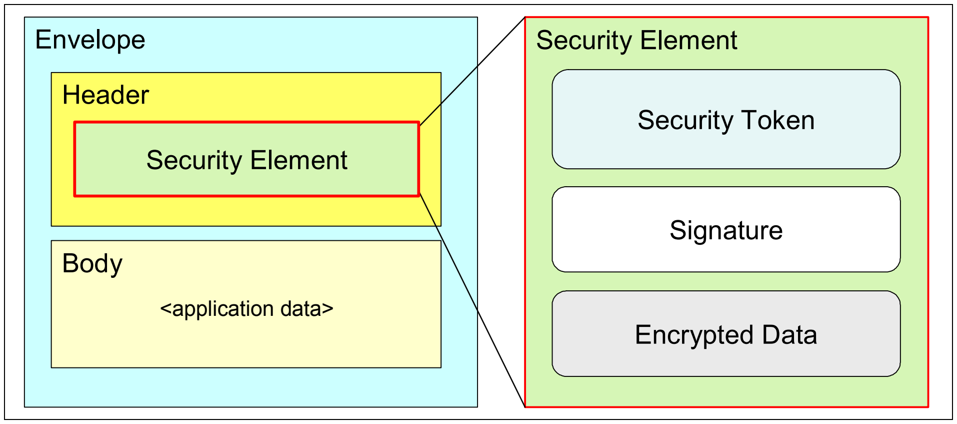


*Figure 2-23 WS-Security: SOAP message security, extensions to the header*

These elements can be used to achieve *message-based security* for a SOAP message. That is, the message itself is tamper-proof and confidential. The origin of the message is provided by the token element. Any change to the message will cause the signature validation to fail so content integrity is provided. An observer of the message cannot read it if it is encrypted, providing message privacy.

Referring to 2.1.1, “Federation example” on page 28, when ABCRBTelco securely passes the client identity and attribute information to ABCRBStocks, the request will use Web services security management on the outbound side.

A SAML assertion is added as a security token in the Web services request and then signing and encrypting it.

This approach allows for the request to be honored by the federated Web service hosted at ABCRBStocks by having the token processed by ABCRBStocks, including the verification, user ID and attribute mapping, authorization, and token transformation that is associated with being a security token consumer.

## 2.5 Web services security management

Web services security management functionality allows the establishment and management of federation relationships for the *active client* scenario. In an active client scenario, an active client, such as an application, is able to generate a Web services request. This request can then be secured (encrypted and signed) to provide message-level confidentiality and integrity. Web services security management adds the ability for message-level authentication, identification, and authorization, in the context of a federation relationship. Web services security management also adds the benefits of the Tivoli Federated Identity Manager trust service, including token services, identity services, and key services.

Web services security management layers over existing WS-Security functionality, providing a WS-Trust (standards-based) approach to the management of security tokens used for authentication purposes within a secured Web services request.

Figure 2-24 on page 82 shows the communications and exchanges that take place at each layer of Tivoli Federated Identity Manager when performing Web services security management.

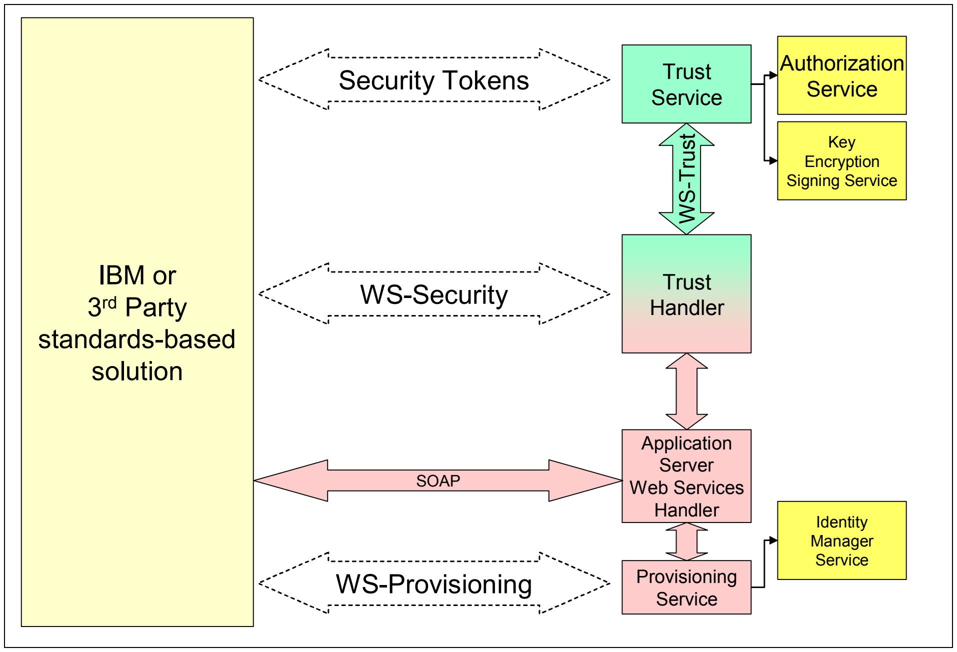


*Figure 2-24 Web services security: Components and communication*

Note that no internal details are shown for the third-party side because their architecture is not known (and not important). Integration is at a protocol level.

At the Communication layer, SOAP messages are being handled by the application server, in this case WebSphere Application Server or WebSphere Web Services Gateway. All real communication is through the Web services handlers in the application server. This component could just as easily be a third-party vendor XML firewall or gateway that has the ability to act as a trust client to the Tivoli Federated Identity Manager trust service.

At the Protocol layer, the WS-Security header in the SOAP request is handled by the Tivoli Federated Identity Manager trust handler (or the third-party XML FW/GW Trust Client). It must read the WS-Security headers sent by the third-party solution (incoming) or include headers for the third-party solution (outgoing).

At the Trust layer, security tokens are being exchanged between Tivoli Federated Identity Manager and the third-party solution. In Tivoli Federated Identity Manager, this layer is handled by the trust service. The trust service exchanges security tokens with the third-party solution in the WS-Security header of SOAP requests (as handled by the trust handler).

## 2.5.1 Architecture overview

Figure 2-25 shows the components required for Web services security management with Tivoli Federated Identity Manager.

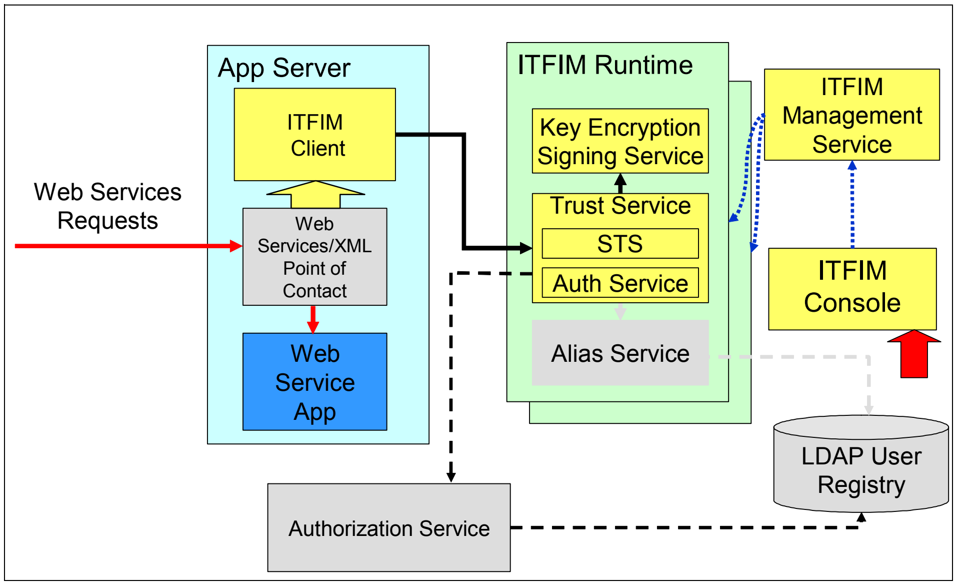


*Figure 2-25 Components for Web services security management*

The Tivoli Federated Identity Manager Web services Trust Client is called by the application server Web services handler during processing of Web services requests. This is triggered by entries in the application's deployment descriptors. The Trust Client builds a WS-Trust based request to the trust service based on the information contained in the Web services request. The trust service will validate existing security tokens and generate new security tokens as required.

In addition to validating incoming security tokens, the trust service may also optionally invoke the authorization service. This authorization decision is used to determine if the identity claimed (and mapped) from the incoming token is allowed to invoke the requested Web services as defined by the WSDL abstract binding.

Assuming the incoming security token is valid and the authorization is successful, the Tivoli Federated Identity Manager Trust Client passes control back to the Web services handler. The Trust Client also passes back identity information that is used to populate the subject associated with the request for J2EE security within the application server.

Figure 2-26 shows a user at company A accessing a resource at company B through a Web service request.

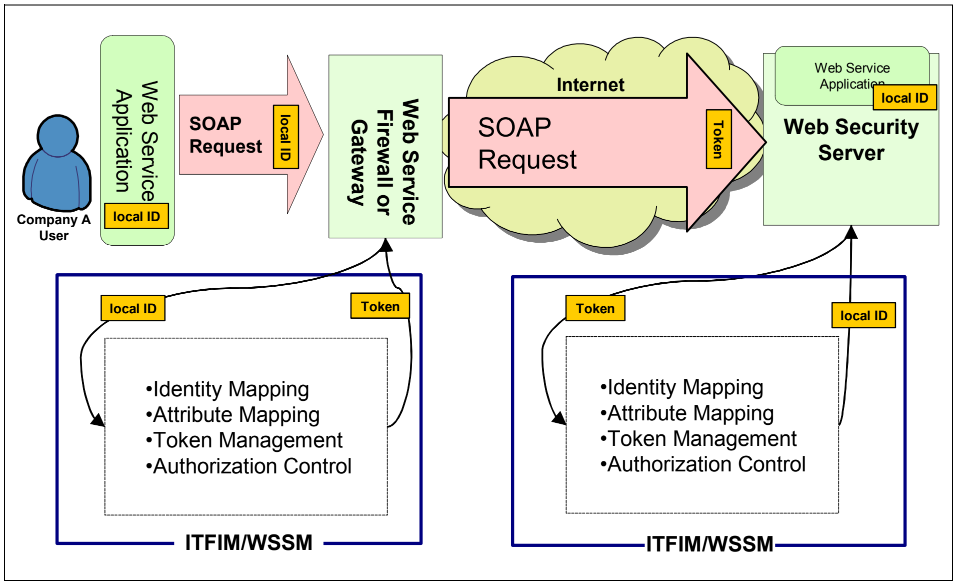


*Figure 2-26 Web service security management (WSSM): Solution architecture*

In the figure:

1. User at company A invokes a Web service using a local ID.
2. The edge of company A could be an XML/WS firewall or gateway or similar. The general requirement for this node is to *standardize* outbound requests so that they can be processed by the receiving company B. Its functionality can include:
   - Mapping of identity claimed in incoming locally valid ID to a token
   - Mapping of local valid attributes such as groups or roles to agreed attributes
   - Exchange of presented local valid token for a token format agreed in the relationship to company B
3. Over the Internet, various technologies can be used to provide message privacy and integrity (SSL, SOAP-Security, VPN tunnel, and so on).
4. Web services functionality at company B side perform authorization and identity/attribute mapping as part of creating a local ID token to be added to the request. The request invokes the backend application as a Web service or as a local application (for example, J2EE or .NET).

To understand the Web services security management solution, it is necessary to explain WS-Security, WS-Trust, and the high-level functionality of a Web services firewall/gateway component, and the Tivoli Federated Identity Manager authorization service. We do this in the next sections.

## 2.5.2 WS-Security

WS-Security is used to accomplish end-to-end message security. Security that is message-based does not rely on secure transport because:

- The message itself is encrypted (message privacy)
- The message itself is signed (message integrity)
- The message contains user identity (proof of origin)

Figure 2-27 illustrates end-to-end message security.

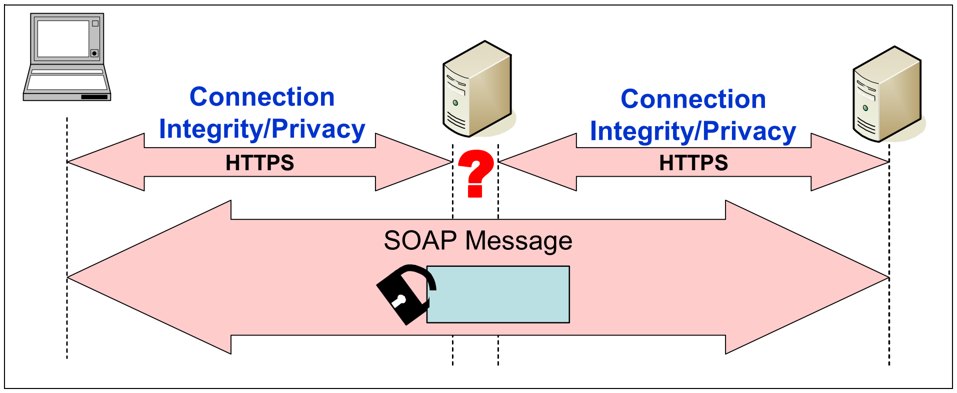


*Figure 2-27 Message-based security: End-to-end security*

The lock on the SOAP message is meant to imply that the SOAP message is inherently secure in and of itself. The SOAP message can be transported in any way and its security is not affected. The SOAP message could be sent as an e-mail attachment, carried on a floppy-disk, and so on, and the properties of privacy, integrity, and proof of origin are not affected.

In contrast, the security of a message that relies on transport security is exposed when that transport security has gaps, as would occur when multiple SSL hops are required to move the message from the origin to the ultimate receiver.

The gaps in the transport security may or may not be an issue, depending on the trust assigned to the nodes that provide the transport compared to the trust required for the message.

For more on the topic WS-Security and SOAP header extensions, see 2.4.6, “Web services security” on page 80.

The elements are defined in the OASIS standard “Web services Security: SOAP Message Security 1.0” and provide the ability to achieve “message-based security” for a SOAP message. That is, the message itself is tamper-proof and confidential.

## 2.5.3 Web services gateway or firewall

A Web services gateway or firewall acts very similar to an HTTP reverse proxy. A WS gateway enables the company to separate internal network topology from the Internet, allowing for flexibility and abstraction. See Figure 2-28.

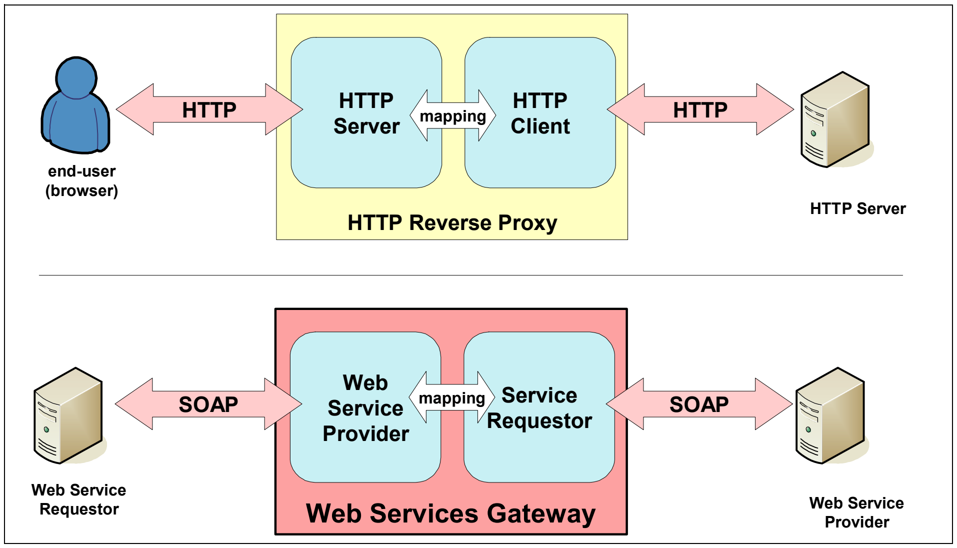


*Figure 2-28 Web services gateway: A reverse-proxy for Web services*

Challenges that are addressed by the Web services gateway are:

- Decoupling of deployment from invocation

  Separating the actual implementation of a service from how another service accesses it enables us to provide process abstraction and flexibility in a deployment scenario:

  – Process abstraction

    The service invocation approach must be flexible enough to cope with events such as switching frequently between external providers of a similar service without requiring changes to the application.

  – Flexibility

    As a service provider, you have the flexibility of changing your deployment infrastructure without notifying all the service requestors. For example, let us say that a Web service is deployed in a machine that later fails during operation. A process should exist to route the invocations to an alternate service in your infrastructure.

- Protocol transformation

  An enterprise may be using a specific messaging infrastructure within their network to meet the business requirements. However, your partners and customers might be using different protocols to invoke your Web service. You must have a mechanism to reconcile the different service invocations to match the requirements of the internal infrastructure.

For more details about the IBM Web Services Gateway see:

http://www.ibm.com/developerworks/webservices/library/ws-gateway/

The SOAP processing model assumes that messages can be relayed among several intermediate SOAP nodes as it travels from the initial sender to the ultimate receiver. The general idea is that, within an enterprise or value chain, intermediaries can handle common aspects of SOAP message processing, thereby leaving the initial sender and ultimate receiver to be concerned only with the behavior required for a particular application.

The IBM Web Services Gateway is a SOAP processing engine that is focused on the operation of the intermediaries in the SOAP chain. Typically, it does not act as an ultimate receiver or as an initial sender of SOAP messages; rather, it can process SOAP messages with the capability to:

- Alter the destination of a message (routing).
- Handle custom header tag processing.
- Apply and remove message level security (WS-Security).
- Perform protocol transformation, for example, submit incoming SOAP/HTTP messages to SOAP/JMS.

For details about Web services gateway see *Federated Identity Management and Web Services Security with IBM Tivoli Security Solutions*, SG24-6394.

### 2.5.4 WS-Trust

The WS-Trust specification defines the interface used to manage the security tokens defined by the WS-Security specification. The Tivoli Federated Identity Manager trust service interface is defined by WS-Trust. It may be accessed by trust clients using either SOAP requests or direct JAVA API calls. The trust client can be the one in Web services security management, SPS, or a custom client, as long as it conforms to the Tivoli Federated Identity Manager WS-Trust profile. This interface allows any conferment Trust Client to request security tokens from the Tivoli Federated Identity Manager trust service, where the trust service can provide the appropriate token translation, identity translation, and request authorization as part of its token functionality. For more information about the trust service refer to “Trust services” on page 74.

## 2.5.5 Authorization services

When used within the context of Web services security management, the trust service can be configured with authorization services (AS). The authorization services may be used to determine if a user (as validated and identified by the trust service) is authorized to access requested resources. This approach allows an implementation-independent decision on the access of a Web service; that is, it does not matter if the Web service exposes a J2EE-based resource, an IBM CICS® resource, or some other proprietary resource.

## 2.5.6 Web services security management architecture approach

Many ways exist to deploy a Web services security management solution. This view gives an attempt to show how it could be accomplished by using Tivoli Federated Identity Manager based nodes, using a Web service gateway. In Figure 2-29 the selected nodes and their connections are represented to illustrate their place meant in the logical network zones.

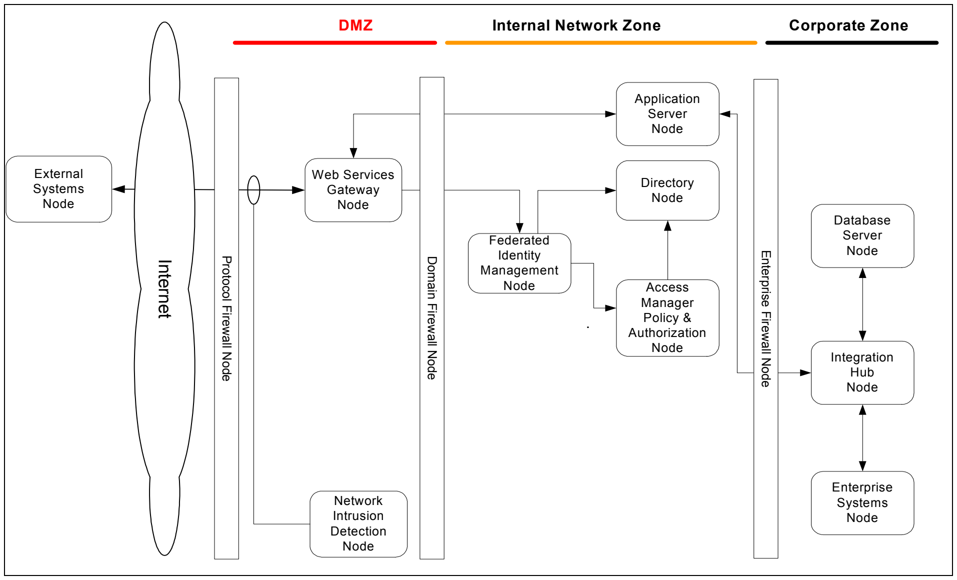


*Figure 2-29 Specified level view of Web services security management*

A more detailed look at Web services security management deployment is available in section “Web services security management” on page 134.

## 2.6 Conclusion

Tivoli Federated Identity Manager relies on many fundamental concepts and knowledge of various standards. Planning an environment involves the discussion of requirements with business partners, and architecting a solution that adheres to agreements. Based on these discussions an architect can then provide a solution that defines roles, attributes, federation protocols, and the necessary components for a Tivoli Federated Identity Manager setup.

Federation standards used in identity protocols must be fully understood in respect to various features so that each may provide for a given requirement. Typically, the simplest protocol is sufficient in most cases. However, more complex protocols, such as SAML 2.0 and Liberty, can provide a more feature-rich standard that can accommodate any requirement.

Tivoli Federated Identity Manager is an ideal product to provide F-SSO services, provisioning services, and Web services security management for an all-in-one solution. In the next chapter we discuss the installation, deployment, and configuration of Tivoli Federated Identity Manager.

3

# Installation and configuration

In this chapter, we discuss the installation and configuration of Tivoli Federated Identity Manager components. The Tivoli Federated Identity Manager product itself is rather small and self-contained within a WebSphere Application Server framework. WebSphere Application Server is a prerequisite of Tivoli Federated Identity Manager. Therefore, WebSphere Application Server skills are critical when deploying Tivoli Federated Identity Manager.

Also, in most environments, Tivoli Federated Identity Manager depend on a front end *point of contact* for session management, and a registry for user management. In Tivoli Federated Identity Manager, Tivoli Access Manager for e-business, and Tivoli Directory Server provide session and registry management respectively. In addition, we also discuss Tivoli Directory Integrator, which is required for provisioning solutions, and the Common Auditing and Reporting Service (CARS), which can fulfill your auditing requirements.

## 3.1 WebSphere Application Server

Several varieties of WebSphere Application Server can be deployed, depending on requirements. Tivoli Federated Identity Manager supports the following varieties of WebSphere Application Server:

- WebSphere Application Server standalone server
- WebSphere Application Server Network Deployment

### 3.1.1 WebSphere Application Server standalone server

WebSphere Application Server can be configured and deployed on a single server. Management functionality is built into the server itself and can be accessed through a Web-based administration console or a command-line interface. WebSphere Application Server requires moderate hardware and can still be deployed when budget resources are limited. Full details about various features can be found in the following WebSphere Application Server 6.0 page:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.base.doc/info/welcome_base.html`

### 3.1.2 WebSphere Application Server Network Deployment

WebSphere Application Server Network Deployment is built upon a redundancy design in order to provide failover and load balancing capabilities. A WebSphere Application Server Network Deployment implementation can span multiple physical servers. Tivoli Federated Identity Manager can take advantage of that redundancy by componentizing Tivoli Federated Identity Manager into management and runtime services. The Tivoli Federated Identity Manager components are discussed in more detail in 3.2, “Tivoli Federated Identity Manager installation” on page 97.

Major components in WebSphere Application Server Network Deployment include a central administration manager named *deployment manager* and *node agents*. Figure 3-1 on page 93 depicts a typical WebSphere Application Server Network Deployment setup.

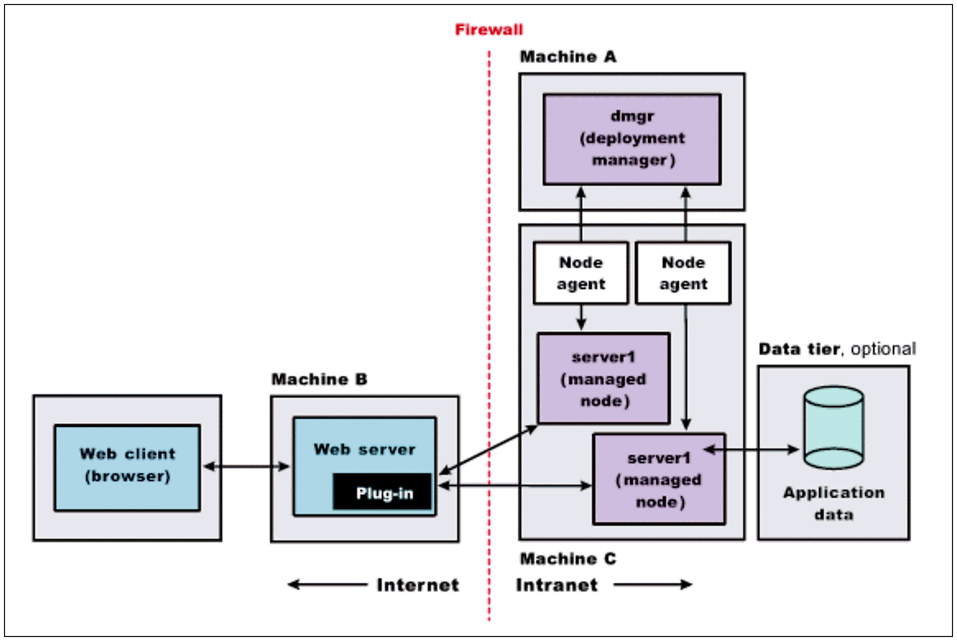


*Figure 3-1 Typical WebSphere Application Server Network Deployment setup*

## Deployment manager

The deployment manager is a central server that provides administration for the WebSphere Application Server Network Deployment *cell*. A cell can contain multiple single application servers, *node agents*, and *clusters*. Administration tasks can be executed through both command-line and Web-based interfaces. The deployment manager is also considered the master repository for the configuration files. Therefore, it ensures that the managed node's configuration files are synchronized with the master configuration repository. Managed nodes or node agents within a WebSphere Application Server Network Deployment cell communicate with the deployment manager for various tasks, as discussed in the next section.

## Node agent

When a node is added to the WebSphere Application Server Network Deployment cell, a node agent server is created. A node agent server manages the servers on that node. Therefore, it monitors the application server's status in addition to maintaining administrative requests to the servers, such as file transfer services, configuration synchronization, and performance monitoring.

### Clustering

To provide failover and load balancing capabilities, WebSphere Application Server Network Deployment provides the concept of *clustering*. Clustering is the grouping of application servers such that enterprise applications are replicated between all servers within the cluster. This grouping provides a more efficient means of administrating application replication.

Tivoli Federated Identity Manager can take advantage of cluster features, thereby allowing Tivoli Federated Identity Manager to scale its payload of requests, and ensuring continuous availability. Configuring Tivoli Federated Identity Manager for clustering is discussed in 3.2, "Tivoli Federated Identity Manager installation" on page 97.

## 3.1.3 Other WebSphere Application Server components

Two additional WebSphere Application Server components are available that we want to examine: the service integration bus and the Web services gateway.

Web services can use the WebSphere Application Server *service integration bus* and the *Web services gateway* to provide a single point of control, access, and validation of Web service requests and allow control of Web services that are available to different groups of Web service users.

Bus-enabled Web services provide a choice of quality of service and message distribution options for Web services, along with intelligence in the form of mediations that allow for the rerouting of messages. The Web services gateway is used to map Web services for use within your organization and by external users, and to manage the relationships between externally-provided Web services and those provided directly through a service integration bus (that is, the relationships between inbound and outbound services).

### WebSphere Application Server service integration bus

With bus-enabled Web services you can achieve the following goals:

- Create an inbound service: Take an internally-hosted service that is available at a bus destination, and make it available as a Web service.
- Create an outbound service: Take an externally-hosted Web service, and make it available internally at a bus destination.
- Create a gateway service: Use the Web services gateway to map an existing service, either an inbound or an outbound service, to a new Web service that appears to be provided by the gateway.

For details about WebSphere Application Server service integration bus Web services, go to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.pmc.nd.multiplatform.doc/tasks/tjw_ep.html

### *Planning for and installing the service integration bus*

Consider how your environment will be configured to support the Web services enablement of the service integration bus. Determine which of the bus-enabled Web services roles you want each server or cluster to perform.

Figure 3-2 shows the main component types and flows for bus-enabled Web services.

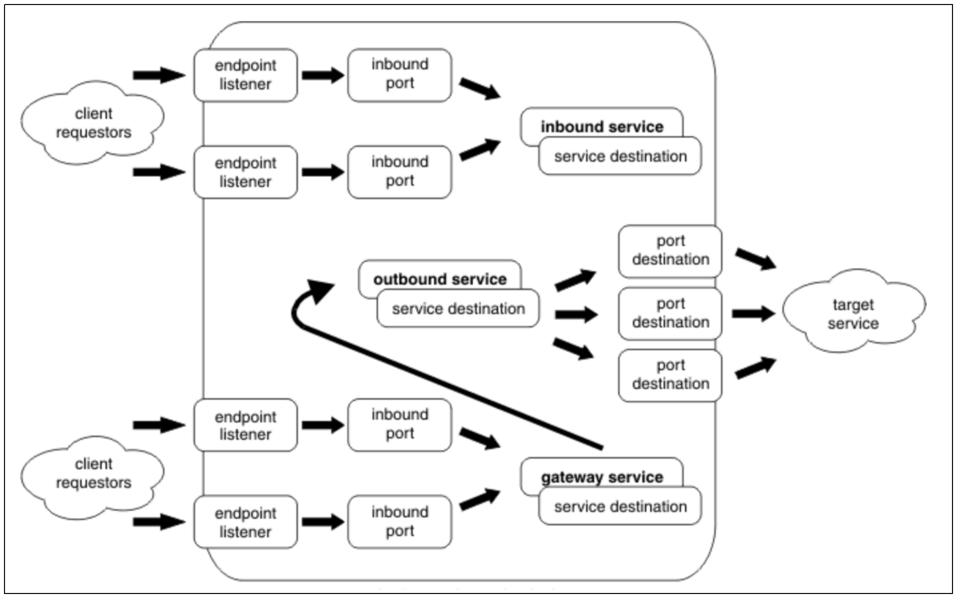


*Figure 3-2 Service integration technologies*

Of all these component types, only three interact directly with the world outside the bus:

- Endpoint listeners
- Outbound ports (which act as service invokers)
- Service destinations (which provide mediation points)

Install and configure, in one or more application servers or clusters, a selection of the service integration bus (SIBus) Web services applications and resources that are copied to your file system by the WebSphere Application Server installation program.

Before you begin, determine the roles that each server or cluster is to perform. Every standalone server or cluster that is to play an SIBus Web services role must be a member of a service integration bus. For more information, refer to the information about configuring the members of a bus at the following WebSphere Application Server 6.0 information center:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.pmc.doc/tasks/tjj0006_.html`

The following steps provide a brief overview of the install procedure:

1. Install and configure the Service Data Objects (SDO) repository.
2. Use the `sibwsInstall.jacl` script to install the service integration technologies resource adapter.
3. Use the `sibwsInstall.jacl` script to install the SIBus Web services application and (optionally) one or more endpoint listener applications.

   If you want to create inbound services, install at least one endpoint listener. This step includes the option to modify the endpoint listener application EAR[1] file and change the default security role that is used to restrict access to the listener.

More details about the installation are available at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.pmc.nd.multiplatform.doc/tasks/tjw_install.html`

### WebSphere Application Server Web services gateway

The *WebSphere Application Server Web services gateway* provides a single point of control, access, and validation of Web service requests. It also allows control of services that are available to different groups of Web service users. Essentially, a WebSphere Application Server Web services gateway instance is used to make available controlled sets of Web services for use within an organization and by external users. The services that each gateway instance makes available as Web services can be a mixture of internal services that are directly available at service integration bus destinations and external Web services.

[1] An Enterprise ARchive, or EAR, is the file format used for packaging application modules into a single archive so that the deployment onto a Web application server can happen simultaneously. It also contains XML files called deployment descriptors that describe how to deploy the modules.

This approach provides the following benefits:

- The gateway service is made available at a different Web address to the target service, so you can replace or relocate the target service without changing the details for the associated gateway service.
- You can have more than one target service (that is, more than one implementation of the same logical service) for each gateway service.
- The gateway service can be made available on a different service integration bus to the target service.
- The gateway provides a common interface to the services in each set. Your gateway service users do not have to know where each underlying service is located, or whether the underlying service is being provided internally or sourced externally, or whether multiple target services are available for a single gateway service.

For more details about how to set up and configure the WebSphere Application Server Web services gateway, go to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.pmc.nd.multiplatform.doc/tasks/twsg_ep.html

## 3.2 Tivoli Federated Identity Manager installation

Tivoli Federated Identity Manager consists of the following components:

- Integrated Solutions Console
- Tivoli Federated Identity Manager management and runtime services
- Web Security Services Manager

For complete details about the Tivoli Federated Identity Manager installation, refer to The following link to the *IBM Tivoli Federated Identity Manager Installation Guide Version 6.1*, GC32-1667, which is available at:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tivoli.fim.doc_6.1/tfim61_install.htm

**Important:** The installation and configuration steps outlined in this section refer to Tivoli Federated Identity Manager 6.1.

### 3.2.1 Integrated Solutions Console

The *Integrated Solutions Console* (ISC) component involves the installation of embedded versions of WebSphere Application Server and WebSphere Portal. The Integrated Solutions Console installation wizard can help you install both WebSphere Application Server Embedded Express 6.0.2 and WebSphere Portal 5.1.0.2 without interruption.

When installing the Integration Solutions Console, user registry entries must already be configured for the administrator. One of the following user registries can be selected during installation:

- IBM Cloudscape

  The IBM Cloudscape database is provided as part of the ISC installation. Therefore, pre-configuration tasks are not required.

- IBM Tivoli Directory Server

  Tivoli Directory Server is not part of the installation, and must be configured beforehand. For Tivoli Federated Identity Manager, an LDAP server should already be installed and configured.

The console prerequisite configuration tasks vary for each user registry type.

**Note:** The Integrated Solutions Console support for LDAP user registries is limited to IBM Tivoli Directory Server 5.1, 5.2, 6.0, and IBM Cloudscape. Tivoli Directory Server 6.0 is provided with Tivoli Federated Identity Manager.

When you choose to use an LDAP user registry, the ISC installation requires that an LDAP server be installed, configured, and running. In addition, the ISC installation requires that several LDAP user registry entries be created prior to starting the installation:

- An Integrated Solutions Console administrative user, such as `iscadmin`.
- An administrative group for Integrated Solutions Console administrators, such as `iscadmins`.
- A user identity for the Integrated Solutions Console server process, such as `iscserver`.

Tivoli Federated Identity Manager provides a program that creates these user registry entries for you. The program uses a defined suffix and provides the necessary LDAP DNs, as follows:

- The entries created by the program match the default user registry entries that are provided during the Integrated Solutions Console graphical installation. If you run the program, and use the default values, you can simply accept the defaults for all values, with the optional exception of the password for the `iscadmin` account. This value defaults to *passw0rd*.
- Optionally, you can customize the user registry entries. If you customize the entries, you will not be able to use the default entries that are provided during the installation.

  See the `ldapconfig.properties` reference at:

  `http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tivoli.fim.doc_6.1/tfim61_install66.htm`

The following methods are for configuring the LDAP user registry for the Integrated Solutions Console. Details and instructions about these methods can be found in the online documentation also.

- Use the `tfimcfg` tool to configure the LDAP user registry

  Reference to `tfimcfg` is located at:

  `http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tivoli.fim.doc_6.1/tfim61_install81.htm`
- Manually configure the LDAP user registry.
- Use an alternate LDAP hierarchy during the installation of ISC.

### 3.2.2 Management services

Let us look at considerations when installing management and runtime services into an existing WebSphere Application Server environment:

- Global security settings
- WebSphere Application Server install location
- SOAP port

*Global security settings* is a WebSphere Application Server configuration that enables authentication requirements when accessing both administration tools and runtime requests. For administration, global security enablement requires users to authenticate into Web-based administration menus. A user registry has to be set up using any of the supported registries. An LDAP server, for example Tivoli Directory Server, or using the internal WebSphere Application Server Cloudspace database, are just some examples. For details about global security enablement within the WebSphere Application Server Network Deployment setup, go to:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tsec_secsetupenable.html`

WebSphere Application Server uses the SOAP port for administration tasks, for both command-line and Web-based client requests. During installation, the Tivoli Federated Identity Manager installation uses the SOAP port to install the management service into the WebSphere Application Server deployment manager (if using WebSphere Application Server Network Deployment) or application server (if using WebSphere Application Server). After the management service is installed, deploying the runtime service is the next step.

## 3.2.3 Runtime services

Deploying Tivoli Federated Identity Manager runtime services requires a domain to be set up first. After setup, the runtime service can then be deployed into an application server or cluster environment. After deployment, configuration of the runtime service is required.

### Domain setup

A domain is a deployment of the Tivoli Federated Identity Manager runtime component to either a single WebSphere Application Server or a WebSphere cluster. Each domain has its own configuration of federations, partners, and mapping rules. These configurations also reference token module instances and encryption keys/certificates.

When Tivoli Federated Identity Manager management service is installed, no domains exist. Using the ISC to create a domain, the following inputs are available:

- WebSphere Application Server global security settings
- Host name and SOAP port of management service
- Tivoli Access Manager for e-business inputs

After the domain is created, the management service can use the input to deploy and configure the runtime service.

### Runtime deployment

The steps for deploying the runtime into a single server environment or into a WebSphere Application Server cluster are described in the online documentation. However, for clustering setups, additional steps are still required (refer to “Cluster steps” on page 102). Other deployments depend on runtime deployment action, such as deploying custom trust modules (that is, plug-ins).

To force a runtime deployment, several steps are required:

1. Update the `serialId` property to a different value. This property is named `com.tivoli.am.fim.rte.software.serialId` and is located in the `<ITFIM_install_root>/pkg/software.properties` file.

   Refer to Example 3-1 to see the property *before* the update.

   *Example 3-1 Before the update*

   ```
   com.tivoli.am.fim.rte.earPropertiesLocation=release/itfim_ear.properties
   com.tivoli.am.fim.rte.publishDirs=plugins:pages
   com.tivoli.am.fim.rte.software.displayName=6.1.0 [2005-07-01T00:01:41Z]
   com.tivoli.am.fim.rte.software.serialId=1120258563124
   com.tivoli.am.fim.rte.installedAt=2005-07-01T22:56:03Z
   ```

   Refer to Example 3-2 to see the property *after* the update.

   *Example 3-2 After the update*

   ```
   com.tivoli.am.fim.rte.earPropertiesLocation=release/itfim_ear.properties
   com.tivoli.am.fim.rte.publishDirs=plugins:pages
   com.tivoli.am.fim.rte.software.displayName=6.1.0 [2005-07-01T00:01:41Z]
   com.tivoli.am.fim.rte.software.serialId=1120258563125
   com.tivoli.am.fim.rte.installedAt=2005-07-01T22:56:03Z
   ```

2. In the Tivoli Federated Identity Manager console, reopen the Runtime Node Management portlet. After the portlet is reopened (or opened) the portlet determines if the `software.properties serialId` has been updated. If so, the console displays an alert message to click the **Deploy Runtime** button.
3. Click the **Deploy Runtime** button.
4. After the new runtime has been deployed, click **Restart WebSphere**.

### *Custom properties*

Within the runtime node management configuration, certain properties are allowed to be customized in order to meet certain requirements within a Tivoli Federated Identity Manager deployment. For most environments, the default values should be adequate. However, the customizable properties are:

- General properties
- Custom properties for:
  - Single sign-on protocol service
  - The alias service
  - The trust service
  - The key service
  - A SOAP client

For a custom properties reference, go to:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tivoli.fim.doc_6.1/tfim61_cfg111.htm

#### Cluster steps

When the Tivoli Federated Identity Manager runtime service is deployed, it is automatically mapped to the default WebSphere Application Server. In clustered environments, the application server is usually deployed into a topology with a Web server, such as IBM HTTP Web server. In this case, a WebSphere plug-in has been installed and configured for the IBM HTTP Web server.

The IBM HTTP Web server performs the workload balancing across cluster members. This means that the Tivoli Federated Identity Manager runtime must be mapped to the Web server.

#### Runtime configuration

This task configures the Tivoli Federated Identity Manager installation runtime into a Tivoli Access Manager for e-business environment. A server identity is created for the Tivoli Federated Identity Manager application. The identity is added into the user registry suffix area where application server identities are defined by Tivoli Access Manager for e-business. This process is the same set of configuration steps taken by Tivoli Access Manager for e-business administrators who use *PDJrte* and *SvrSslCfg* to add applications to a Tivoli Access Manager for e-business domain. For Tivoli Federated Identity Manager runtime configuration, inputs are actually taken from Tivoli Access Manager for e-business settings within the Tivoli Federated Identity Manager domain properties.

## 3.3 Tivoli Federated Identity Manager configuration

After Tivoli Federated Identity Manager runtime service has been successfully deployed, an administrator can proceed in configuring federations. Configuring federations should only take place after agreements have been drafted with business partners. Various F-SSO parameters or Web services protocols have to be defined.

### 3.3.1 Federated single sign-on

Tivoli Federated Identity Manager separates federated single sign-on (F-SSO) configuration into *federations* and *partners*. Partner configuration, associated with the federations, allows custom modifications of the F-SSO protocol, which can then better adhere to each individual partner agreement.

## 3.3.2 Federations

A single sign-on federation is a trust relationship between two or more business partners. The trust relationship is built based on agreements that have been reached between the business partners. The agreements include both the definition of specific business policies and the adoption of agreed-upon technical standards. When the agreements have been defined, the administrator builds a trust infrastructure to support them. The building of a trust infrastructure requires definition of a number of technical concepts, such as:

- Identity provider and service provider roles
- Point-of-contact server
- Federation protocol profiles
- Security token module instance
- Identity mapping
- Encryption keys

Table 3-1 highlights several characteristics of each protocol: SAML 1.0, 1.1, and 2.0 (OASIS standards), Liberty ID-FF 1.0, 1.1,and 1.2 (Liberty Alliance published specifications), and WS-Federation (WS-Fed) Passive (IBM, Microsoft, RSA, VeriSign published specification).

*Table 3-1 Characteristics for each SSO protocol*

| **Supported characteristic** | **SAML 1.0, 1.1** | **SAML 2.0** | **Liberty ID-FF 1.0, 1.1, 1.2** | **WS-Federation** |
|---|---|---|---|---|
| PUSH SSO: Identity provider (IdP) initiated SSO | Yes[a] | Yes | No[b] | Yes |
| PULL SSO: Service provider (SP) initiated SSO | Yes | Yes | Yes | Yes |
| Front channel security token exchange | Yes | Yes | Yes | Yes |
| Back channel security token exchange | Yes | Yes | Yes | No[c] |
| Choice of security token type | No | No | No | Yes |
| Where are you from? (WAYF) support | N/A | Yes | Yes | Yes |
| Accounts at IdP and SP are required to initiate SSO | Yes[d] | Yes | Yes[d] | No |

| Supported characteristic | SAML 1.0, 1.1 | SAML 2.0 | Liberty ID-FF 1.0, 1.1, 1.2 | WS-Federation |
|---|---|---|---|---|
| Accounts at IdP and SP are required to initiate account linking process | N/A | Yes | Yes | No |
| IdP-initiated account linking (federation) within SSO process | No | Yes | No | Yes |
| SP-initiated account linking (federation) within SSO process | No | Yes | Yes | Yes |
| Support for single logout (SLO) or single sign-off | No | Yes | Yes | Yes |
| Create account on SP-side as part of IdP-initiated SSO or account linking: just in time provisioning (JITP) | Yes | Yes | No | Yes |
| Account de-linking where user had pre-existing accounts before account linking | Yes | Yes | Yes | Yes |
| Account de-linking where user did not have pre-existing accounts before account linking | Yes[e,f] | Yes | Yes[e,f] | Yes[f] |

a. Although not explicitly part of SAML, this can be implemented by a vendor. This type of implementation will almost certainly break cross-vendor interoperability.

b. This is not part of the Liberty ID-FF conformance profile. This can be implemented by a vendor, but will almost certainly break cross-vendor interoperability.

c. The WS-Federation Passive scenario used to demonstrate interoperability employed a front-channel token exchange. Back-channel exchange can be supported using a direct trust server to trust server security token request, replacing the information passed in the front channel with an artifact-type security token.

d. The profiles for SAML and Liberty ID-FF explicitly require accounts at both the IdP and SP side as a prerequisite for SSO and account linking. A particular vendor implementation might not require this (in the table, see the supported characteristic “IdP-initiated account linking (federation) within SSO process” for more details).

e. This is somewhat out of the scope of SAML and Liberty ID-FF implementation, as they both require that a user had accounts at both sides before the account linking process was initiated.

f. Assuming that the SP side account was created in response to runtime provisioning, this account must have been created in a manner that allows it to be converted from an SSO account to a direct-authentication account.

Using Table 3-1 as a reference, and using agreed-upon requirements with a business partner, the properties of the federation can be created. Those include the following details:

- Federation properties
- WS-Federation properties
- SAML 1 properties
- SAML 2 profiles and properties

## Common federation properties

In Table 3-2, we list common properties for a typical federation.

*Table 3-2 Common federation properties*

| **Property** | **Description** |
|---|---|
| Federation name | The federation should have a unique name. The name can be any character string. Using the administration console, assign a string that describes the purpose of the federation. For example, if the federation is between a business and the health benefits administration for the employees of the business, the name of the federation might be: *Health Benefits Provider*<br><br>This name enables the administrator to easily select this federation (from the console menu choices) when future management actions are required to update the configuration of the federation. |
| Role | Identity Provider or Server Provider |
| Protocol | ► Liberty ID-FF 1.1<br>► Liberty ID-FF 1.2<br>► SAML 1.0<br>► SAML 1.1<br>► SAML 2.0<br>► WS-Federation Passive Profile |
| Identity mapping properties | XSLT code or custom module |
| Contact information | The Tivoli Federated Identity Manager federation wizard allows you to enter contact information for your business partner.<br><br>The company name is required. The other values are optional.<br><br>This information is not used by the Tivoli Federated Identity Manager software. It exists primarily as a place to record useful reference information. When you need to supply your business partner with information about your federation, as part of the process for adding a partner, you can reference this contact information. |

## WS-Federation properties

Table 3-3 lists the WS-Federation properties required during configuration.

*Table 3-3 WS-Federation properties*

| Property | Description |
|---|---|
| Number of seconds before the issue date that an assertion is considered valid | An integer value that specifies the number of seconds that the assertion is considered valid. Specified during token configuration on identity provider only. Default value is 60 seconds. No minimum or maximum is enforced. |
| Number of seconds that the assertion is valid after being issued | An integer value that specifies the number of seconds that the assertion remains valid. Specified during token configuration on identity provider only. The default value is 60 seconds. No minimum or maximum is enforced. |
| WS-Federation Realm | The name of the WS-Federation Realm. This name is the unique identifier for this instance of Tivoli Federated Identity Manager. The Realm name is included in assertions that are sent to federation business partners. Partners rely on finding a known (defined) Realm name in order to accept the assertions. A default value is provided. For example:<br><br>`https://idp.example.com/FIM/sps/wsfed/wsf`<br><br>The string `wsfed` is the name of the federation. The endpoint is automatically created. Accept the default name. |
| WS-Federation Endpoint | The endpoint for all requests for WS-Federation services. A default value is provided. For example:<br><br>`https://idp.ibm.com/FIM/sps/wsfed/wsf`<br><br>The string `wsfed` is the name of the federation. The endpoint is automatically created. Accept the default name. |

## SAML 1 properties

Table 3-4 lists the SAML 1 properties.

*Table 3-4 SAML 1 properties*

| Property | Description |
|---|---|
| Number of seconds before the issue date that an assertion is considered valid | Specified during token configuration on identity provider only. Default value is 60 seconds. No minimum or maximum is enforced. |
| Number of seconds that the assertion is valid after being issued | An integer value that specifies the number of seconds that the assertion remains valid. Specified during token configuration on identity provider only. The default value is 60 seconds. No minimum or maximum is enforced. |
| Enable one-time use enforcement | Specifies whether this artifact (token) is to be used only once. Specified during token configuration on service provider only. |
| Provider ID | A unique identifier that identifies the provider to its partner provider. This property is set on both the identity provider and the service provider.<br><br>For example, on a host named `idp.example.com`, with a federation named saml10-idp, the default Provider ID is:<br><br>`https://idp.example.com/FIM/sps/saml10-idp` |
| Source ID | An automatically generated ID based on the Provider ID. This property is set on both the identity provider and the service provider. This value cannot be modified. |
| Intersite Transfer Service URL | The URL to which the Service Provider sends authentication requests. This property is set only on the identity provider. A default value is provided. For example:<br><br>`https://idp.example.com/FIM/sps/myfed/saml/login` |
| Artifact Resolution Service URL | The endpoint for SOAP requests from service providers who are exchanging artifacts for assertions. This property is set only on the identity provider. A default value is provided. For example:<br><br>`https://idp.ibm.com/FIM/sps/myfed/saml/soap` |
| Artifact Cache Lifetime | The number of seconds that an identity provider retains an assertion in the cache for the browser-artifact profile when it sends an artifact to a service provider. If the service provider does not access the assertion in this time period, the identity provider discards the assertion. This property is set only on the identity provider. The default value is 30 seconds. |

| Property | Description |
|---|---|
| Allow IBM PROTOCOL Extension | Enables or disables whether the identity provider permits the use of an IBM PROTOCOL extension. The extension allows a query-string parameter that specifies whether to use browser artifact or browser post. This property is set only on the identity provider. The default value is disabled. |
| Select Signing Key | A key in a known keystore to be used to sign SAML response messages. You can modify this value by typing the name of a different key in the same keystore. The value should take the form KeystoreName_KeyAlias.<br><br>This property is set on the identity provider only. |
| Assertion Consumer Service URL | Endpoint for the single sign-on responses received from the identity provider. This property is set only on the service provider. A default value is provided. For example:<br><br>`https://sp.example.com/FIM/sps/myfed/saml/login` |
| SAML messages for Browser POST are signed as required by the protocol | This property is always enforced because the protocol requires it. It is specified during identity provider configuration only. |
| Sign SAML messages for Artifact profile | This property requires the identity provider to sign messages for the browser artifact profile. It is specified during identity provider configuration only. Selection of this field during the Federation Wizard activates the Key lookup table. When this field is enabled, you must select a signing key from the key table. Default value is enabled. |
| Sign Artifact Requests | Specifies that the service provider will always sign artifact requests. This field is displayed by the Federation Wizard for service providers only. Selection of this field activates the Key lookup table. When this field is enabled, you must select a key from the key table. Default value is disabled. |

## SAML 2 profiles and properties

The SAML 2 protocol supports a large number of features. Therefore, many properties exist when configuring a SAML 2.0 setup. Tivoli Federated Identity Manager supported profiles for SAML 2.0 are described in this section.

### *Web single sign-on*

The Web browser single sign-on profile enables a client using a Web browser to achieve single sign-on access to resources within a SAML 2.0 federation. Typically the user wants to access a resource provided by a service provider, and must authenticate with an identity provider in order to be granted that access. The profile provides a mechanism for the Web user to obtain an authentication assertion that can be used to establish a security context within the federation.

Establishment of the security context enables a user to access multiple resources within the federation without having to authenticate more than once. Provided in Table 3-5, are SAML 2 properties common to SAML protocols and properties specific to SAML 2 standard.

*Table 3-5 SAML 2 common properties and single sign-on properties*

| **Property** | **Description** |
|---|---|
| Provider ID | A unique identifier that identifies the provider. This property is set on both the identity provider and the service provider. For example, on a host named `idp.example.com`, with a federation named saml20-idp, the default Provider ID is:<br><br>`https://idp.example.com/FIM/sps/saml20-idp/saml20` |
| Source ID | A unique identifier based on the Provider ID. The value is automatically generated. You cannot modify the value. |
| SOAP Endpoint URL | The URL of the SOAP endpoint to be used when SOAP binding is configured. It defaults to:<br><br>`https://point_of_contact_server/sps/federation_name/saml20/soap` |
| **Single sign-on** | |
| Single sign-on Service URL | The URL on the identity provider that the service provider uses to send single sign-on requests. The default value provided is:<br><br>`https://point_of_contact_server/sps/federation_name/saml20/login` |
| Assertion Consumer Service URL | The URL at the service provider that an identity provider should use when sending single sign-on or federation responses. The default value is:<br><br>`https://point_of_contact_server/sps/federation_name/saml20/login`<br><br>The property is specified for a service provider federation. |
| Supported Bindings for single sign-on | Three bindings are supported and can be selected:<br>► Browser artifact<br>► Browser POST<br>► Browser Redirect |
| Message Lifetime (seconds) | An integer value specifying the length of time, in seconds, that a message is valid.<br><br>The default is 300 seconds. |

| Property | Description |
|---|---|
| Artifact Lifetime (seconds) | The length of time, in seconds, that an artifact is considered valid. This field is enabled only when browser artifact binding has been enabled.<br><br>The defaults is 120 seconds. |
| Session Timeout (seconds) | The length of time, in seconds, that the session remains valid.<br><br>The defaults is 7200 seconds. |
| Require consent to federate | Requires the identity provider to present a page to the user verifying the federation request.<br><br>This option is specified only when configuring an identity provider federation. |
| Identity provider is allowed, but not required, to interact with the user. | The identity provider can optionally interact with the user when the user is attempting authentication. Specified for a service provider federation only. |
| Single sign-on in Passive | Prevents the identity provider from interacting with the user during authentication. Specified for a service provider federation only. |
| Force Users to Authenticate at Identity Provider | Forces the identity provider to authenticate a user even when the user has previously authenticated. Specified for a service provider federation only. |

### *Enhanced client or proxy profile*

An *enhanced client or proxy* (ECP) is a system entity that communicates with an identity provider and service provider on behalf of a user (client). For example, a user might request a resource from a service provider. The service provider might not know which identity provider to access in order to authenticate the user. The service provider can contact the ECP, which knows how to locate and access the appropriate identity provider. The ECP supports SOAP and reverse SOAP (PAOS) bindings during the processing of authentication requests.

Table 3-6 lists the property for a SAML-enabled client that contains knowledge of a user's identity provider and supports SOAP bindings for enhanced client or proxy configurations. Enhanced client or proxy replaces the *Where Are You From* (WAYF) method of resolving the identity provider for a user.

*Table 3-6 SAML 2 enhanced client or proxy*

| **Property** | **Description** |
|---|---|
| **Enhanced client or proxy** | |
| HTTP Headers | A list of HTTP headers that the identity provider can expect to see from the Enhanced Client Proxy clients containing identity information. Separate each header by a new line character. For example:<br><br>`Header1`<br>`Header2`<br>`Header3` |

### *Identity Provider Discovery*

The *Identity Provider Discovery* profile is used by service providers to discover which identity provider is used by a user (principal) during Web browser single sign-on. Some deployments have more than one identity provider, and the service provider must be able to determine which identity provider a principal uses. The Identity Provider Discovery profile uses a cookie that is created in a domain that is common between identity providers and service providers in a given deployment. The cookie contains the list of identity providers, and is called the *common domain cookie.*

Table 3-7 lists properties allowed for Identity Provider Discovery.

*Table 3-7 SAML 2 Identity Provider Discovery*

| **Property** | **Description** |
|---|---|
| **Identity Provider Discovery** | |
| Common Domain Name | The name of the domain shared with other members of the federation. For example:<br><br>`example.com`<br><br>There is no default value. |
| Common Domain Cookie Service URL | The URL of this provider within the common domain. Must be specified as a URL of the point of contact for the provider and must include the common domain value. For example, for a system named `idp.example.com` and common domain value of `somecommondomain.com`, the value would be:<br><br>`https://idp.somecommondomain.com` |
| Common Domain Cookie Lifetime (seconds) | An integer value for the lifetime, in seconds, of the common domain cookie. Specified only on the identity provider.<br><br>The default value is `-1`, which means that the cookie does not expire.<br><br>No maximum value is imposed. |

### *Name Identifier Management*

The *Name Identifier Management* profile manages user identities that are exchanged between identity providers and service providers. The profile enables identity providers to notify service providers when there is a change to the content or format of an identity for a given user (principal). The profile enables service providers to specify unique aliases for the principal, and to send those aliases to the identity provider to be used instead of the principal name. The profile also enables either provider to notify the partner when the provider decides to no longer issue or accept messages that use the principal's identity.

Table 3-8 lists properties that can be used to configure Name Identifier Management.

*Table 3-8 Name Identifier Management*

| **Property** | **Description** |
|---|---|
| **Name identifier** | |
| Management URL | The URL that the partner contacts to use the Name Identifier Management service.<br><br>For example, on an identity provider host `idp.example.com` with a federation named `saml20_ip`:<br><br>`http://idp.example.com/sps/saml20_ip/mnids`<br><br>For example, on an service provider host `sp.example.com` with a federation named `saml20_sp`:<br><br>`http://sp.example.com/sps/saml20_sp/mnids` |
| Supported Bindings | See supported bindings in “SAML 2 bindings” on page 115. |

### *Single logout*

The *single logout* profile is used to terminate all login sessions that are currently active for a specified user within the federation. A user who achieves single sign-on to a federation establishes sessions with more than one participant in the federation. The sessions are managed by a session authority, which in many cases is an identity provider. When the user wants to end sessions with all session participants, the session authority can use the single logout profile to globally terminate all active sessions.

Table 3-9 lists properties available for single logout.

*Table 3-9 SAML 2 Single logout*

| Property | Description |
|---|---|
| **Single logout** | |
| Single Logout Service URL | The URL that the partner contacts to use the single logout profile.<br><br>For example, on an identity provider host `idp.example.com` with a federation named `saml20_ip`:<br><br>`http://idp.example.com/sps/saml20_ip/slo`<br><br>For example, on an service provider host `sp.example.com` with a federation named `saml20_sp`:<br><br>`http://sp.example.com/sps/saml20_sp/slo` |
| Supported Bindings | See supported bindings in “SAML 2 bindings” on page 115. |

Table 3-10, SAML 2 lists essentially the same signature properties as SAML 1.

*Table 3-10 SAML 2 Signature Options*

| Property | Description |
|---|---|
| **Signature options** | |
| Require Service Provider Sign Authentication Requests | Requires the service provider to sign authentication requests. Specified during identity provider configuration only. The default value is enabled. |
| Require Identity Provider Sign Assertions | Requires the identity provider to sign assertions. Specified during service provider configuration only. The default value is enabled. |
| Sign SAML 2.0 Messages | Specifies that the provider will always sign messages. This field is displayed on the Signatures panel for both identity providers and service providers. Selection of this field activates the Key lookup table. The default value is enabled. |
| Select Signing Key | Specifies the signing key for signing assertions or authentication requests. For example: `DefaultKeyStore_testkey` |

In Table 3-11, SAML 2 provides the method to encrypt messages.

*Table 3-11 SAML 2 Encryption property*

| Property | Description |
|---|---|
| **Encryption property** | |
| Encryption key identifier | Specifies the encryption key for encrypting SAML messages.<br>For example: `DefaultKeyStore_testkey` |

Table 3-12 lists properties that are associated with timeout values for SAML 2 assertions.

*Table 3-12 SAML 2 Token configuration*

| Property | Description |
|---|---|
| **Token configuration** | |
| Number of seconds before the issue date that an assertion is considered valid. | This field must contain a value. The default value is 60 seconds.<br><br>No minimum or maximum value is enforced. |
| Amount of time the assertion is valid after being issued (seconds) | This field must contain a value. The default value is 60 seconds.<br><br>No minimum or maximum value is enforced. |

Table 3-13 lists *identity mapping properties* configurations available for SAML 2, which are not different from SAML 1.

*Table 3-13 SAML 2 Identity Mapping Properties*

| Property | Description |
|---|---|
| **Identity mapping properties** | |
| Identity Mapping Module Instance | The module that specifies mapping rules for identities. |
| Change Identity Mapping Module Instance | Invokes the Identity Mapping Options panel. The Identity Mapping Options panel enables you to select either an XSL transformation for identity mapping or a custom mapping module instance. |

## SAML 2 bindings

Certain SAML 2 properties require knowledge of binds that are required for a particular SAML 2 profile. SAML 2 does provide the flexibility of supporting more than one of type of binding.

#### *Browser Redirect or HTTP Redirect*

*HTTP Redirect* enables SAML protocol messages to be transmitted within URL parameters. It enables SAML requestors and responders to communicate using an HTTP user agent as an intermediary. This might be necessary if the communicating entities do not have a direct path of communication, or if the responder requires interaction with a user agent such as an authentication agent.

HTTP Redirect is sometimes called *Browser Redirect*, particularly when used in single sign-on operations.

This profile is selected by default.

#### *HTTP POST or Browser POST*

*HTTP POST* enables SAML protocol messages to be transmitted within an HTML form using base64-encoded content. It enables SAML requestors and responders to communicate using an HTTP user agent as an intermediary. This might be necessary if the communicating entities do not have a direct path of communication, or if the responder requires interaction with a user agent such as an authentication agent.

HTTP POST is sometimes called *Browser POST*, particularly when used in single sign-on operations. Browser POST uses a self-posting form during the establishment and use of a trusted session between an identity provider, a service provider, and a client (browser).

#### *HTTP Artifact or Browser Artifact*

*HTTP Artifact* is a binding in which a SAML request or response (or both) are transmitted by reference using a small piece of code called an artifact. A separate binding, such as a SOAP binding, is used to exchange the artifact for the actual protocol message. It enables SAML requestors and responders to communicate using an HTTP user agent as an intermediary, but when it is not preferable to expose the message content to the intermediary.

HTTP Artifact is sometimes called *Browser Artifact*, particularly when used in single sign-on operations. Browser Artifact uses a SOAP back channel to exchange an artifact during the establishment and use of a trusted session between an identity provider, a service provider, and a client (browser).

#### *SOAP*

Use SOAP bindings for communication.

### Liberty

Liberty federations use the properties described in “Common federation properties” on page 105 plus the properties listed in Table 3-14 on page 117.

*Table 3-14 Liberty single sign-on federation properties*

| **Property** | **Description** |
|---|---|
| Provider ID | A unique identifier that identifies the provider to its partner provider. This property is set on both the identity provider and the service provider. The value is automatically generated by the administration console. For example:<br><br>`https://idp.example.com/FIM/sps/libertyfed/liberty`<br><br>In the example, the components of the Provider ID are:<br><br>► `https://idp.example.com/FIM/sps/`<br><br>This is the value that you specified on the point-of-contact server configuration screen.<br><br>► `libertyfed`<br><br>This is the name that you specified for the Liberty federation.<br><br>► `liberty`<br><br>This is the defined string that specifies the federation type of Liberty. Do not change this value. |
| Succinct ID | An automatically generated ID that is based on the Provider ID. This property is set on both the identity provider and the service provider. This value cannot be modified. |
| SOAP Endpoint | SOAP endpoint location at the service provider or identity provider to which Liberty SOAP messages are sent.<br><br>This setting is required when one or both of the following conditions is true:<br><br>► Browser artifact single sign-on profile is selected on the Liberty profiles window.<br>► One or more of the optional Liberty profiles are selected and SOAP/HTTP communication initiated by at least one of the service providers is selected.<br><br>For example:<br>`https://idp.example.com/FIM/sps/libertyfed/liberty/soap` |
| Amount of time the assertion is valid after being issued (seconds) | An integer value that specifies the number of seconds that the assertion remains valid. This is specified for Liberty tokens. The minimum value is 120 seconds. The maximum value is 300 seconds. |

| Property | Description |
|---|---|
| Single sign-on Service URL | The URL on the identity provider to which the service provider sends single sign-on and federation requests. This property is set on the identity provider only. A default value is provided. For example:<br><br>`https://idp.example.com/FIM/sps/libertyfed/liberty/login` |
| Liberty Message Lifetime | An integer value indicating the amount of time, in seconds, that a Liberty message remains valid. This property is set on both the identity provider and the service provider.<br><br>The minimum value is 60 seconds. There is no maximum value other than the maximum integer supported by the data type.<br><br>The default value is 60 seconds. |
| Liberty Artifact Lifetime | An integer indicating the time, in seconds, in which a service provider must retrieve an assertion from an identity provider. The service provider uses an artifact to retrieve the assertion. The identity provider keeps the mapping of the artifact to the assertion in its cache for this amount of time. When the service provider does not collect the artifact in this amount of time, the cache purges the artifact and the service provider login fails.<br><br>This property is specified only when configuring an identity provider with browser artifact single sign-on profile.<br><br>**Note:**<br>This value is not used for browser POST profile.<br><br>The minimum value is 120 seconds, which is also the default value. |
| Require Consent to Federate | Enables or disables the requirement that the identity provider prompt the user to consent to joining the federation. This property is set on the identity provider only. This message is presented when federating of the user account occurs. Default value is disabled. Select the check box to enable issuing of the prompt. |
| Assertion Consumer Service URL | The URL on the service provider to which the identity provider sends single sign-on or federation responses. This property is set on the service provider only. A default value is provided. For example:<br><br>`https://sp.example.com/FIM/sps/libertyfed/liberty/login` |
| Single sign-on is Passive (Identity Provider does not interact with user) | Enables or disables the requirement that the identity provider must not interact with principal (user) and must not take control of the user interface from the service provider. This property is set only when configuring a service provider. Select the check box to enable this requirement. The default value is disabled. |

| Property | Description |
|---|---|
| Force Identity Provider to authenticate user | Enables or disables a requirement that the identity provider must authenticate a user (Principal) regardless of whether the user is already authenticated. This value is specified only when the single sign-on is Passive (Identity Provider does not interact with user) check box is cleared. This property is set only when configuring a service provider.<br><br>When this setting is cleared, the identity provider must authenticate the user (principal) only when the user is not presently authenticated. Select the Force Identity Provider check box to authenticate user (to enable this requirement). Clear the check box to disable this requirement. |
| User Browser Artifact Profile for single sign-on | This check box is displayed on the federation properties panel for a Liberty federation on a service provider. The value reflects the choice made when the federations was configured. Browser artifact is one of the single sign-on profiles supported by Liberty. To enable this profile, select User Browser Artifact Profile for single sign-on. To disable this profile, clear the check box. |
| Sign Liberty Messages | Enables or disables the signing of Liberty communications from this provider to federation business partners. This property also specifies whether Liberty tokens will be signed. This property is set on both the identity provider and service provider. Select the check box to enable signing. Clear the check box to disable signing. During federation creation, this check box is selected by default.<br><br>When you select this check box, you must also specify a keystore and a key to use for signing the messages. |
| Signing Key Identifier | The name of the keystore and key file that are used to sign the message or token. For example:<br><br>`DefaultKeyStore_testkey` |

## Federation endpoints

Endpoints are in URL format and denote how the federation exposes various features of the identity protocol.

### *Liberty endpoints*

Depending on which liberty features are used within the federation determines the resulting endpoints. The following types of endpoints can be instantiated after a Liberty federation is created:

- Liberty single sign-on endpoints
  - Endpoint for receiving a single sign-on request message on an identity provider (single sign-on Service URL) is:

    ```
    https://point_of_contact/your_junction/sps/Liberty_federation_nam
    e/liberty/login
    ```

    For example:

    ```
    https://ip.example.com/FIM/sps/libertyfed/liberty/login
    ```

  - Endpoint for forcing authentication

    This endpoint is specified on the identity provider only. This endpoint is not configurable, but must be protected by a Tivoli Access Manager for e-business ACL:

    ```
    https://point_of_contact/your_junction/sps/Liberty_federation_nam
    e/liberty/auth
    ```

    For example:

    ```
    https://ip.example.com/FIM/sps/libertyfed/liberty/auth
    ```

  - Endpoint for receiving a single sign-on request message on a service provider (Assertion Consumer Service URL) is:

    ```
    https://point_of_contact/your_junction/sps/Liberty_federation_nam
    e/liberty/login
    ```

    For example:

    ```
    https://sp.example.com/FIM/sps/libertyfed/liberty/login
    ```

- Liberty single logout endpoints
  - Endpoint for initiating a single logout request from a user agent or for receiving a single logout request message is:

    ```
    https://point_of_contact/your_junction/sps/Liberty_federation_nam
    e/liberty/slo
    ```

    For example:

    ```
    https://ip.example.com/FIM/sps/libertyfed/liberty/slo
    ```

   - Endpoint for receiving a single logout response

     `https://point_of_contact/your_junction/sps/Liberty_federation_name/liberty/sloreturn`

     For example:

     `https://ip.example.com/FIM/sps/libertyfed/liberty/sloreturn`

- Liberty SOAP endpoint

  Endpoint for receiving SOAP requests is used on both identity provider and service provider is:

  `https://point_of_contact/your_junction/sps/Liberty_federation_name/liberty/soap`

  For example, identity provider:

  `https://ip.example.com/FIM/sps/libertyfed/liberty/soap`

  For example, service provider:

  `https://sp.example.com/FIM/sps/libertyfed/liberty/soap`

- Liberty register name identifier (RNI) endpoints

   - Endpoint for initiating an RNI from a user agent is:

     `https://point_of_contact/your_junction/sps/Liberty_federation_name/liberty/rniinitial`

     For example, when initiated from an identity provider:

     `https://ip.example.com/FIM/sps/libertyfed/liberty/rniinitial?ProviderID=https://sp.example.com/FIM/sps/libertyfed-sp/liberty`

     For example, when initiated from a service provider:

     `https://sp.example.com/FIM/sps/libertyfed-sp/liberty/rniinitial?ProviderID=https://ip.example.com/FIM/sps/libertyfed/liberty`

   - Endpoint for receiving an RNI request message is:

     `https://point_of_contact/your_junction/sps/Liberty_federation_name/liberty/rni`

     For example:

     `https://ip.example.com/FIM/sps/libertyfed/liberty/rni`

   - Endpoint for receiving an RNI response is:

     `https://point_of_contact/your_junction/sps/Liberty_federation_name/liberty/rnireturn`

     For example:

     `https://ip.example.com/FIM/sps/libertyfed/liberty/rnireturn`

- Liberty federation termination notification (FTN) endpoints
  - Endpoint for initiating a federation termination notification (FTN) from a user agent is:

    `https://point_of_contact/your_junction/sps/Liberty_federation_name/liberty/ftninitial`

    For example, when initiated from an identity provider:

    `https://ip.example.com/FIM/sps/libertyfed/liberty/ftninitial?ProviderID=https://sp.example.com/FIM/sps/libertyfed-sp/liberty`

    For example, when initiated from a service provider:

    `https://sp.example.com/FIM/sps/libertyfed-sp/liberty/ftninitial?ProviderID=https://ip.example.com/FIM/sps/libertyfed/liberty`
  - Endpoint for receiving an FTN notification message is:

    `https://point_of_contact/your_junction/sps/Liberty_federation_name/liberty/ftn`

    For example:

    `https://ip.example.com/FIM/sps/libertyfed/liberty/ftn`
  - Endpoint for receiving an FTN response is:

    `https://point_of_contact/your_junction/sps/Liberty_federation_name/liberty/ftnreturn`

    For example:

    `https://ip.example.com/FIM/sps/libertyfed/liberty/ftnreturn`

### *SAML 1 endpoints*

SAML 1 endpoints can be of the following two types:

- SAML 1 single sign-on endpoints

  Endpoint to receive a single sign-on request message (both identity provider and service provider) is:

  `https://point_of_contact/your_junction/sps/your_federation_name/saml/login`

  For example:

  `https://ip.example.com/FIM/sps/samlfed/saml/login`

  The SAML 1.0 specification does not specify which profile (browser-artifact or browser-post) that an identity provider should use when communicating with a service provider.

  The Tivoli Federated Identity Manager SAML single sign-on support allows for configuration of the single sign-on profile on a per-partner basis. Additionally, Tivoli Federated Identity Manager supports an optional query-string parameter that specifies which profile that the identity provider should use. This query-string parameter overrides local identity provider configuration. The standard request to the identity provider for initiating single sign-on has the following form:

```
https://ip/intersite_transfer_service?TARGET=https://sp/sso_endpoint
```

- SAML 1 SOAP endpoint

  Endpoint on which to receive SOAP requests:

  ```
  https://point_of_contact/your_junction/sps/your_federation_name/saml/soap
  ```

  For example:

  ```
  https://ip.example.com/FIM/sps/samlfed/saml/soap
  ```

### *SAML 2 endpoints*

For details about SAML 2 endpoints, go to:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tivoli.fim.doc_6.1/tfim61_fsso61.htm

### *WS-Federation endpoints*

WS-Federation single sign-on federations use the same endpoint for single sign-on, single logout, and signout-cleanup:

```
https://point_of_contact/your_junction/sps/your_federation_name/wsf
```

The action that the protocol module must execute is provided as a query string parameter, as follows:

- Single sign-on:

  ```
  wa=wsignin1.0
  ```

- Single logout:

  ```
  wa=wsignout1.0
  ```

  For example, when sign out is initiated from the identity provider:

  ```
  https://ip.example.com/FIM/sps/wsfed/wsf?wa=wsignout1.0
  ```

  For example, when sign out is initiated from the service provider:

  ```
  https://sp.example.com/FIM/sps/wsfed-sp/wsf?wa=wsignout1.0
  ```

- Single logout cleanup:

  ```
  wa=wsignoutcleanup1.0
  ```

## 3.3.3 Federation business partners

A federation consists of a trust relationship between two business partners. After creating a federation adding a business partner to the federation is the next step. The business partner by definition will play the role opposite of the federation. An example is if the federation role is an identity partner, then any partners associated to the federation will be service providers.

Before adding a business partner, configuration information must first be obtained from the partner. The processes for exchanging information are described in the following topics:

- Obtaining federation configuration data from the partner
- Obtaining requirements for SOAP endpoint access
- Providing federation properties to partner

### Obtaining federation configuration data from the partner

For Liberty and SAML 2 federations, the partner exports the federation configuration to a metadata file. The metadata file can then be imported into your Liberty or SAML 2 federation within the partner configuration wizard. For SAML 1, the partner can either export the federation configuration to a metadata file or manually communicate the federation configuration. For WS-Federations, the partner must manually communicate the federation configuration. After the partner's configuration information is received, the partner wizard can then be used to add the partner's federation properties. Note the following information:

- For WS-Federations, the partner administrator must manually compile the necessary information. The partner must provide this information before using the console to add the partner. After received, the partner information can the be manually entered when prompted by the Partner Wizard.

  For SAML 1 federations, the same process applies when the partner chooses not to provide a metadata file.

  For SAML federations, certificate information might also be required. See “Obtaining requirements for SOAP endpoint access” on page 125.

- For Liberty and SAML federations, certificate information might also be required for SOAP endpoint access.

  When importing metadata into a Liberty or SAML 2.0 federation, the administration console takes the certificates that are included in the metadata and imports them into the managed keystores. If the partner name used when importing the metadata file is the same as a name used during a previous importation of metadata, the administration console imports the key (certificate) into the same alias.

In most cases, this causes expected behavior because the same metadata file is being re-imported from an existing partner. However, administrators who choose to reuse a partner name for other purposes should be aware that during metadata importation the existing keys (certificates) that are stored under the partner's alias in the keystore are overwritten.

The administration console can also be used to modify partner properties after the partner has been added. For example, a partner might have to supply an updated key or certificate to replace one that is due to expire. Updating the appropriate partner properties to reflect the use of the new key, would be done within the federation business partner menu.

## Obtaining requirements for SOAP endpoint access

For Liberty and SAML federations, authentication information might also have to be known (certificates and basic authentication information) for use with SOAP endpoints.

Before adding a partner, determine with the partner whether SSL (HTTPS) is used to protect the SOAP back channel. The SOAP back channel is used with browser artifact single sign-on profile. It is also used with SAML 2.0 profiles that support SOAP binding, and optionally can be used with Liberty profiles such as RNI, FTN, and SLO.

When the partner uses HTTPS for the SOAP channel, the partner must:

- Provide the validation certificate that will be used to validate SSL communication from partner site, when partner-initiated messages are received at the local federation SOAP endpoint.
- Determine whether client certificate authentication is required when contacting the SOAP endpoint. When the partner does require client certificate authentication, a decision must be made about what certificate will be presented, when establishing the SSL session. The choice of certificate is a business decision. The certificate can be one already owned or one that the partner provides. The Tivoli Federated Identity Manager key service can be used to import agreed upon certificates.
- Determine whether basic authentication is required when establishing a connection to the SOAP endpoint. When the partner requires basic authentication, the partner must supply the user name and password to present, in order to establish an authenticated session.

**Note:** When the partner requires client authentication, the partner must specify either client certificate authentication or basic authentication. Only one form of authentication can be specified within Tivoli Federated Identity Manager partner configuration.

### Providing federation properties to partner

Steps taken to provide necessary information and data are specific to whether a metadata file or a manual collection is required, as follows:

- Liberty and SAML 2 federations
  a. Use the administration console to generate a metadata file that contains the necessary federation configuration.
  b. If required, provide the partner with certificates for use with SSL communication to SOAP endpoints. If basic authentication is also required, you must provide a user name and password.
- SAML 1 federations

  The option of manually collecting the necessary configuration instead of exporting the properties to a file is possible.

  **Note:** Use of a metadata file is good practice. It eliminates the chance of errors being made during the manual input of data. Support for manual collection of information is provided for backward compatibility only.

  If collecting information manually is required, complete the following tasks:
  a. Use the administration console Federation Properties panel to obtain the properties. Use the contents of the Federation Properties panel to guide you to the properties that apply to your federation.
  b. If required for SAML federations, provide the partner with certificates for use with SSL communication to SOAP endpoints. If basic authentication is also required, you must provide a user name and password.
- WS-Federation single sign-on federations.

  Manually collect information and data by using the administration console Federation Properties panel to obtain the properties.

## 3.3.4 Identity mapping

A primary function of the trust service is to provide the security tokens necessary for token exchange between partners in a federation. The exchange requires more than just the conversion of a proprietary token to the type of token that the partner is expecting. The contents in the token must also be built and modified according to the expectations of the partner.

Two partners in a federation can each use the same token standard but can maintain different information within the token. For example, each partner might format a user account number in a different way, and might store that piece of information in a different attribute within the security token.

The trust service provides a mechanism for accommodating these differences in both user account information and the use of attributes within tokens. The mechanism is an identity mapping module that executes according to an identity mapping rule that has been defined for each federation business partnership. The trust service mapping module performs the mapping and looks up the configured identity mapping for the specified partner.

For example, mapping rules can specify how to:

- Map incoming identities to local accounts.
- Assign a user ID based on an arbitrary attribute within the token.
- Map user attributes that have different names, such as an e-mail address.

Information from an incoming token can be manipulated and mapped to an outgoing token in whatever manner is required. For example, specific values can be inserted into the token, and using Java programs (or JavaScript) to acquire information from external sources is also possible. The robust handling of identity mapping rules is achieved by using eXstensible Stylesheet Language (XSL) transformations for identity mapping.

To accomplish this mapping, the trust service has defined an internal XML format for storing information. This format is independent of any specific token format, and can be described as *token-neutral*.

The use of this format enables the trust service modules to:

1. Convert the contents of a received token into a token-neutral XML representation of the data.
2. Perform the necessary information mapping onto the XML file.
3. Convert the modified contents from the token-neutral XML format into the format required by the output token.

Therefore, in a standard single sign-on trust chain, the incoming token is converted to this neutral format, the identity mapping is performed, and then the outgoing token is created.

Figure 3-3 shows the sequence of processing for trust chains that are created for F-SSO operations.

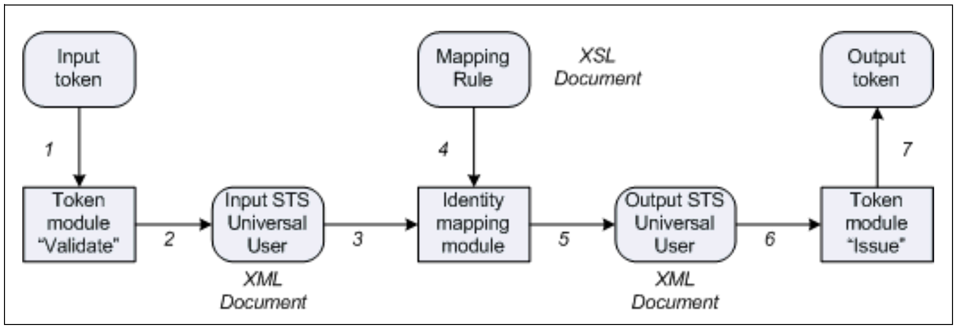


*Figure 3-3 Sequence of processing by a trust module chain*

This type of trust chain is created automatically when the administrator configures F-SSO through the Tivoli Federated Identity Manager console Federation wizard, as follows:

1. The token module, operating in validate mode, receives the input token and validates it.
2. The token module converts the contents of the input token into an Input Security Token Service (STS) universal user XML document.
3. The identity mapping module reads the Input STS universal user document as a starting point for the Output XML document.
4. The identity mapping module modifies the contents of the Input STS universal user to reflect the structure that is required by the output token module. The identity mapping module also consults an identity mapping rule file (XSL document) to determine the correct data values and content for the Output STS universal user XML document.
5. The identity mapping module creates the Output STS universal user XML document.
6. The token module, operating in *exchange* or *issue* mode, reads the Output STS universal user XML document as input for the output token conversion process.
7. The token module issues (creates) the output token in the format that is required by the single sign-on federation business partner.

## Security Token Service universal user XML documents

The trust service uses a token-neutral format consisting of an XML document called the *Security Token Service (STS) universal user*. The selection of XML for the neutral format allows the use of XSL to specify identity mapping rules. The STS universal user is an XML document that can contain three sections:

- Principal information
- Group information
- Attribute information

The contents of the STS universal user document are generic, in order to accommodate the varying requirements of the different security token formats that must be supported. The exact information contained in any specific Input STS universal user document is dependent on the token type for the security token that was used as input. The information required in an output STS universal user document depends on:

- The token type to be generated
- The specific mapping rule being used for the conversion

To understand how to use XSL mapping rules to create the XML files, you must understand the basic structure of the STS universal user XML documents.

The XML namespace of the STS universal user element is:

```
xmlns:stsuser="urn:ibm:names:ITFIM:1.0:stsuser
```

The prefix `stsuser` is used in the following descriptions of the STS universal user elements:

- The STS universal user document contains an element for storing information about the user (principal) identity:

  ```
  <stsuser:Principal>
  ```

- The STS universal user document can contain an element for storing information about the group memberships:

  ```
  <stsuser:GroupList>
  ```

- The STS universal user document typically contains an element for storing a list of attributes:

  ```
  <stsuser:AttributeList>
  ```

Within each element (`Principal`, `GroupList`, `AttributeList`) are a number of `<stsuser:Attribute>` elements. Each of the `Attribute` elements has additional elements:

- Name

  This stores the name of the attribute
- Type

  This stores a type for the name. Typically this specifies the expected format.
- Value

  Each *<stsuuser:Value>* element contains a value for the Attribute.

STS universal user XML documents can be obtained by turning on Tivoli Federated Identity Manager *tracing* for the trust service.

## Use of XSL language for creating mapping rules files

The identity mapping module uses the Java API for XML Parsing (JAXP) to transform the input document based on XSL configuration that you specify in an XSL file.

XSL is a language that can be used to transform (format) documents. XSL is used to define stylesheets for HTML and to format XML data so that it can be displayed in a Web browser. Part of the XSL standard defines transformations for moving data from one form to another. The transformation language can include conditional statements, variables, and call-outs to Java programs.

The trust service uses XSL as a language to create mapping rules that specify how to transform an input STS universal user document into an output STS universal user document that can be used to generate an output token. The XSL parser processes XSL documents by looking for matching templates. When a template is found, the contents of the template are processed.

The main tasks that are performed in mapping rules are to:

- Move identity information between elements.
- Reformat existing identity information.
- Add new elements with new identity information.
- Remove unwanted identity information.

The IBM Rational Application Developer tool set can be used to run an XSL debugger from a command line. This tool enables testing of XSL code without having to run the trust service.

### Custom mapping module

Rather than use XSLT to form a mapping code, Tivoli Federated Identity Manager also allows the use of a *custom mapping module* that uses Java and its openness to solve any mapping requirements. Rational Application Developer or an Eclipse development environment can be used to code a mapping module. After the code is ready to be deployed, the module must be inserted into a new directory within the plug-ins directory (`<TFIM_install_root>/plugins`), and deployed to the Tivoli Federated Identity Manager runtime service. See "Runtime deployment" on page 100.

After runtime service is deployed, execute the following steps:

1. Create module type by selecting the newly deployed module.
2. Create module instance of the new module type.

## 3.3.5 Alias service

*Alias service* is used when federations of SAML 2.0 and Liberty are required. Tivoli Federated Identity Manager 6.1 uses LDAP to store alias'. Therefore, an existing LDAP server needs to be configured and available before configuring alias services. The following list describes the required steps to configure Tivoli Federated Identity Manager the alias service. After setting the alias service properties, you are prompted to restart WebSphere Application Server.

- Creating an LDAP suffix for the alias service

  You must create an LDAP suffix `cn=itfim` to enable the alias service to access the LDAP user registry:

  a. Stop the LDAP server:

     In UNIX, issue the following command:

     ```
     # ibmdirctl -D cn=root -w passw0rd stop
     ```

     In Windows, use the Services icon.

  b. Add the suffix using the following LDAP command:

     ```
        # idscfgsuf -s "cn=itfim"
     ```

  c. Start the IBM LDAP server.

     In UNIX, issue the following command:

     ```
     # ibmdirctl -D cn=root -w passw0rd start
     ```

     In Windows, use the Services icon.

d. Use the **`ldapmodify`** command to update the LDAP schema file. For example, in UNIX or Linux:

   - IBM Tivoli Directory server:

     ```
     ldapmodify -D cn=root -W passw0rd -f
     /opt/IBM/FIM/etc/itfim-secuser.ldif
     ```

   - Sun™ ONE Directory server:

     ```
     ldapmodify -D cn=root -W passw0rd -f
     /opt/IBM/FIM/etc/itfim-secuser-sunone.ldif
     ```

- Configuring LDAP server settings

  The alias service communicates with the user registry server (LDAP) to manipulate user identity information. The alias service must be configured with the correct LDAP settings. Follow these steps:

  a. Select **Tivoli Federated Identity Manager** → **Domain Management** → **Alias Service Settings**. The Alias Service Settings panel is displayed.

  b. Specify the properties for the alias service to use when searching the LDAP user registry.

     See Configuring LDAP Search Settings at:

     http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tivoli.fim.doc_6.1/tfim61_fsso85.htm

  c. Specify communication properties for the alias service to use when communicating with LDAP servers.

     See Configuring LDAP Environment properties at:

     http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tivoli.fim.doc_6.1/tfim61_fsso85.htm

  d. Specify configuration parameters for each LDAP server.

     See Configuring LDAP Server properties at:

     http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tivoli.fim.doc_6.1/tfim61_fsso85.htm

  e. Choose one of the following actions:

     - Click **Apply** to save configuration properties without exiting from the window.
     - Click **OK** to save configuration properties and exit from the window.

- Configuration actions

  – Specify LDAP Host search order

    The LDAP Hosts box lists the configured servers in order of preference. The alias service tries first to contact the server at the start (top) of the list.

  If that contact is unsuccessful, the alias service attempts to contact the next server in the list.

  Use the up and down arrows located on the right side of the box to move individual LDAP servers higher or lower in the order of priority.

- Add

  Click **Add** to activate the LDAP configuration fields for the selected server. This selection enables you to enter values into the configuration fields for a new LDAP server.

- Remove

  Click **Remove** to delete an LDAP server from the LDAP hosts configuration list. You must select a server before clicking **Remove**.

- Save

  Click **Save** to save the LDAP properties that you entered into the configuration fields for a server. When you save the properties, the console inserts the host name and port number into the LDAP hosts box.

- Reset

  Click **Reset** to reset the LDAP configuration fields to their default values.

- Configuration properties

  - LDAP Hostname

    The fully qualified name of the LDAP server. For example:

    ```
    idp.example.com
    ```

  - Port

    This configures the port on which the LDAP server listens. The default port for non-SSL communication is:

    ```
    389
    ```

    The default port for SSL communication is:

    ```
    636
    ```

  - Bind DN

    The Bind DB is the distinguished name (DN) that the alias service uses to bind to the LDAP server. The default value is:

    ```
    cn=root
    ```

  - Bind Password

    This is the password for the DN specified in the Bind DN field.

- Key Name

  The name of the encryption key to use when establishing SSL communication. Select a key name from the list of names. The names in the list are obtained from the keystore that is specified in the Keystore field in the LDAP environment portion of this configuration window.
- Minimum number of connections

  The initial number of connections (binds) for the alias service to establish to the LDAP server. The minimum valid number is zero (0). The maximum valid number is limited only by the maximum value supported by the data type.

  The default value is 2. Use the default value unless you have a specific reason to increase it.
- Maximum number of connections

  The maximum number of connections (binds) for the alias service to establish to the LDAP server. The maximum valid number is limited only by the maximum value supported by the data type.

  The default value is 10. Use the default value unless you have a specific reason to increase it.

### 3.3.6 Web services security management

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

The Web services security management component provides functions for:

- Web service providers, which have to process inbound security tokens using a token consumer
- Web service clients, which have to create outbound security tokens using a token generator

The processing and creation of the tokens is performed 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.

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 deployment of a F-SSO environment. Unlike the other Tivoli Federated Identity Manager components, the Web services security management component does not require the use of the 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, this feature permits a SAML assertion to be directly used for authentication.
- Permits the type of token to be exchanged; for example, a `UsernameToken` can be exchanged for a SAML assertion.
- Permits authorization checks to be made on a Web service before invoking the Web service.
- Permits the user identity to be mapped; for example, a user identity can be mapped to a local identity expected by the Web service.
- Permits both many-to-one and one-to-many user identity mappings.
- Permits authorization checks to be made on a Web service before invoking the Web service. Using the capabilities provided by Tivoli Access Manager for e-business, requests can be validated without actually invoking the underlying Web service.
- All of these features are available at both the Web service client and the Web service.

## Preparing WSSM for a Web service application

To prepare WSSM for a Web service application, follow these steps.

1. Install the WSSM component as described in the *IBM Tivoli Federated Identity Manager Installation Guide Version 6.1.1*, GC32-1667.
2. Perform configuration tasks for the component and its required software, including:
   a. Configuring WebSphere Application Server. Details about WebSphere Application server configuration for WSSM component are provided at:

      `http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tivoli.fim.doc_6.1/tfim610_wssm_guide40.htm`

   b. Updating configuration batch and script files. Details are provided at:

      `http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tivoli.fim.doc_6.1/tfim610_wssm_guide53.htm`

   c. Configuring Tivoli Access Manager for e-business, if you plan to use the authorization function of the Web services security management component.
   d. Configuring keystores and certificates, if keys or certificates will be used with the tokens you plan to validate or exchange.

## Preparing the Web service application to use WSSM

The general steps involved to enable a Web service application to be used with the Web services security management component are as follows:

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

   This process is described in the following areas:

   – The component summary describes the component parts, their class names, and parameters. Refer to “Configuration of a Web services application” on page 142 for the list of required components.

   – The configuration information for how your Web services application uses Web services security management is described in the application's deployment descriptors. This step is necessary for both generating and consuming tokens. For more information refer to “Configuration of a Web services application” on page 142.

2. Configure the Tivoli Federated Identity Manager trust service to be used with your Web service application. Details about enabling application security, which consists of configuring an application and a partner chain, are at:

   http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tivoli.fim.doc_6.1/tfim610_wssm_guide71.htm

In “Trust service” on page 136, we provide more details about the trust service configuration.

## Trust service

When used with the WSSM, the Tivoli Federated Identity Manager trust service can manage the handling of security tokens that are part of the requests made to Web service applications.

The trust service processes requests based on the identity of the requestor (who is making the request) and the destination of the request (what is being requested). To convey this information, the request is structured according to the WS-Trust standard format that is defined for the Security Token Requests.

The request includes defined fields such as *Issuer* and *AppliesTo*:

- The *Issuer* is the issuer of the request. In the WSSM component, the issuer is either the WSSM token generator or the WSSM token consumer. These components are listed in “Configuration of a Web services application” on page 142.
- The *AppliesTo* field identifies the Web service that is the destination of the request.

The trust service uses the Issuer and the AppliesTo fields, along with other requests for specific WS-Trust functions, in order to determine how to process the request.

To process tokens, the trust service uses:

- Modules
- Module instances
- Module chains

The general process used by the trust service is as follows:

1. Receive the RequestSecurityToken.
2. Determine which module chain should be used to process the request.
3. Invoke, in order, each module in the chain.
4. Use the resulting data to create a RequestSecurityTokenResponse.
5. Return the RequestSecurityTokenResponse.

### *Modules*

The Tivoli Federated Identity Manager trust service is designed to accept plug-ins that handle the different token format conventions. These plug-ins are called *modules*.

For each type of token, the trust service has a security token module that handles the trust relationship. These modules are responsible for creating tokens, validating tokens, and exchanging token types. The exchange of token types enables the passing of trust relationship information between different security domains.

The modules provide the following functions:

- Process incoming tokens (SAML, username, and so on) to validate the signature and to verify that the token has not expired.
- Create tokens to pass user authentication information to partners, based on the local user authentication token that was passed in.

- Create Tivoli Access Manager for e-business credentials for local authentication, based on the authentication identity received from a federated partner.
- Perform identity mapping as part of exchanging token types.
- Perform authorization as part of a Web services security management deployment.

The Web services security management component supports the following token module types by default:

- SAML10STSModule (SAML 1.0)

  Provides support for creating or consuming SAML 1.0 tokens.
- SAML11STSModule (SAML 1.1)

  Provides support for creating or consuming SAML 1.1 tokens.
- SAML20STSModule (SAML 2.0)

  Provides support for creating or consuming SAML 2.0 tokens.
- UsernameTokenSTSModule (Username Token)

  Provides support for creating or consuming 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.
- KerberosSTSModule

  Provides support for consuming 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 consuming PassTicket tokens. In Resource Access Control Facility (IBM RACF®) secured sign-on, 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 rather than sending a RACF password across the network.
- X509STSModule

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

The WSSM component supports the following processing module types by default. These modules can be added to a module chain to perform specialized processing:

- DynamicChainSelectionModule

  This module requests that the trust service invoke 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 F-SSO.

- XSLTransformationModule

  This module calls an XSL parser to read identity mapping rules in order to generate a Secure Token Service (STS) Universal User XML document that contains user identity information.

- AuthorizationSTSModule

  This module interacts with the authorization engine, provided by Tivoli Access Manager, to execute an authorization check for the supplied user identity. This module is used by only the Web services security management component and requires the use and configuration of Tivoli Access Manager.

Other module types supported by Tivoli Federated Identity Manager in single sign-on federations, such as JAASModule, IVCredModule, Liberty11STSModule, Liberty12STSModule, DelegationModule, and DSigModule are not supported by the Web services security management component.

### *Module instances*

A *module instance* is a combination of a module with optional parameters that provide instructions for specific processing of the module. Multiple module instances can be defined using each module type.

### *Module chains*

*Module chains* (also called *trust chains*) are groups of module instances that are configured for use together.

The trust service calls each module instance in the chain sequentially to perform a specific function as part of the overall processing of a request. Information is passed between the module instances in the chain until the required result is obtained or the request is rejected.

To allow different communities of users to access the same Web service but use different token types or different token processing requirements, the module chains must provide for the requirements of the community (the partner) and the requirements of the Web service application.

Therefore, to create a complete module chain for use with the Web services security management component, you must create and configure two types of chains:

- Partner chain

  The partner chain is capable of processing the token submitted by the partner, including token validation and possibly mapping logic that is unique to that partner.

- Web service application chain

  The Web service application configuration creates a trust module chain that is capable of exchanging or issuing a security token to use for authentication to the Web service application, including identity mapping, attribute management, and Tivoli Access Manager authorization.

### Partner chain

A partner chain is partner-specific and is configured to handle the specific type of token that is sent by a partner. A partner chain is one of the two chains that are necessary to create a complete chain for use with the Web services security management component. Figure 3-4 shows an example of a partner module chain in the Web services security management component.

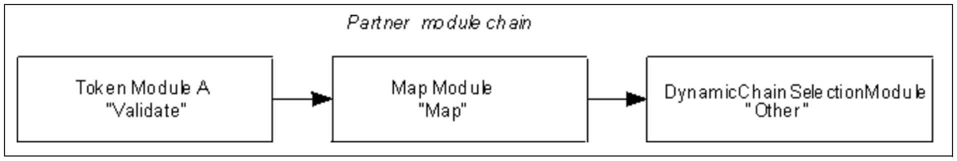

Partner module chain
Token Module A "Validate"
Map Module "Map"
DynamicChainSelectionModule "Other"

*Figure 3-4 Typical partner module chain*

The primary sequence for the tasks in a partner chain is Validate → Map → Other (Link), as follows:

1. *Validate* an incoming token from the partner, and convert the data to a token-neutral format. The module used is one of the supported token modules (such as UsernameSTSModule).
2. *Map* and modify the user identity data within the token-neutral format to reflect the application-specific requirements. (Mapping is optional. The XSLTransformationModule is provided with Web services security management component for mapping.) Also, see 3.3.4, “Identity mapping” on page 126.
3. *Other* is a module mode that executes application-specific processing. In the case of a partner chain, the DynamicChainSelectionModule is used to link the partner chain to the application chain.

### *Web service application chain*

A Web service application chain is specific to a Web service application and is configured to handle the requirements of the application. A Web service application chain is one of the two chains that are needed to create a complete chain for use with the Web services security management component.
Figure 3-5 shows an application module chain that, typically, consists of three modules instances.

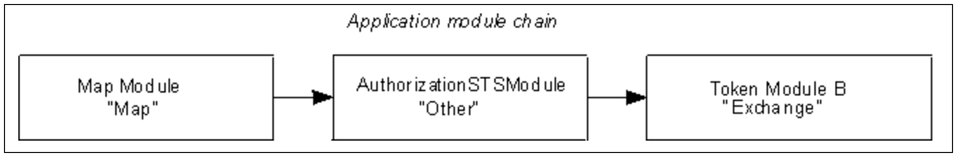


*Figure 3-5 Typical application module chain*

The primary sequence for the tasks in an application chain is Map → Other (Authorization check) → Exchange, as follows:

1. *Map* and modify the user identity data within the token-neutral format to reflect the application-specific requirements. (Mapping is optional. The XSLTransformationModule is provided with Web services security management component for mapping.)
2. *Other* is a module mode that executes application-specific processing. In the case of an application chain, the AuthorizationSTSModule can be used to perform an authorization check for the supplied user identity. (The use of the authorization module is optional. It requires the use of a Tivoli Access Manager for e-business environment).
3. *Exchange* or issue an incoming token to the appropriate token type for authentication to the Web service application. The module used is one of the supported token modules (such as the SAML11STSModule).

### *Module chain example*

The partner chain and the application chain are used in combination to create a complete module chain. For example, you might have a partner that sends username tokens and a Web service application that requires SAML 1.1 tokens.

The module chain to support this scenario could be configured as follows:

- Partner chain:
  - Token Module A

    Username token module (UsernameTokenModule) in Validate mode
  - Map Module

    No mapping module used
  - *Other* Module

    Dynamic chain selection module (DynamicChainSelectionModule) in *Other* mode
- Web service application chain:
  - Map Module

    Identity mapping module (XSLTransformationModule) in Map mode
  - *Other* Module

    Authorization module (AuthorizationModule) in *Other* mode
  - Token Module B

    SAML 1.1 module (SAML11Module) in Exchange mode

## Configuration of a Web services application

Steps required for creating a connection between a previously created and configured Web service application, and the Web services security management component, require knowledge of the components involved. The following components are involved in configuring WSSM:

- Token consumers
- Login modules
- WSSM token generator
- Callback handlers

Details about each of the components can be found at:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tivoli.fim.doc_6.1/tfim610_wssm_guide29.htm`

### *JAAS login security token*

The *security token* that is used to perform a JAAS login can be made available to the Web service application by the Web services security management component. This token is accessed through the following object that is a private credential in the JAAS Subject object:

```
com.tivoli.am.fim.wssm.credentials.WSSMCredential
```

The `WSSMCredential` object includes a getXmlSecurityToken method that returns the security token as a string. Access to this token from the Web service application might be useful if, for example, the application processes attribute statements in a SAML assertion token.

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();
String securityToken = wssmCredential.getXmlSecurityToken();
```

## 3.3.7 Federated identity services

Provisioning is a key element in identity management for enterprises. Account provisioning is triggered within a company's internal trust domain when there is a change in account status for a user. For example, a new user account must be created when a new employee is hired; or, user account permissions might require modification when an employee changes job roles.

Enterprise provisioning systems such as Tivoli Federated Identity Manager provide the functions necessary to evaluate role-based provisioning policy, and can create or modify multiple user accounts as appropriate. Management of these accounts includes management of account IDs and passwords in order to assure secure authenticated communication within the enterprise.

Federated identity provisioning extends these provisioning management activities beyond an internal trust domain. Federated identity provisioning makes possible the extending of local account provisioning at an identity provider to include federated account provisioning out to multiple service provider partners. A service provider, when notified of the federated provisioning request, can perform the local provisioning necessary to supply its service to the specified employee.

Provisioning requests sent between identity providers and service providers must be secure and must be based on open standards. Tivoli Federated Identity Manager satisfies these requirements by providing an implementation of the *WS-Provisioning* standard.

WS-Provisioning is a specification authored by IBM to provide a Web service interface to communicate provisioning requests and responses. It includes operations for adding, modifying, deleting, and querying provisioning data. It also specifies a notification interface for subscribing to provisioning events. Provisioning data is described using XML and other types of schema. This facilitates the translation of data between different provisioning systems.

WS-Provisioning is part of the service-oriented architecture and has been submitted to the Organization for the Advancement of Structured Information Standards (OASIS) Provisioning Service Technical Committee. Tivoli Federated Identity Manager 6.1 supports draft version 0.7 of the WS-Provisioning specification. The WS-Provisioning interface is an open standard that is available to other companies that want to develop interoperable provisioning scenarios and systems.

The Tivoli Federated Identity Manager provisioning components are:

- The Tivoli Federated Identity Manager WS-Provisioning Web service

  This service runs as an application on WebSphere Application Server Version 6.0.

- WS-Provisioning connectors

  The connectors run on Tivoli Directory Integrator. See 3.4, “Tivoli Directory Integrator” on page 149 for more detail.

Figure 3-6 shows the high-level sequence of actions that occur when a client (identity provider) sends a provisioning event to a server (service provider):

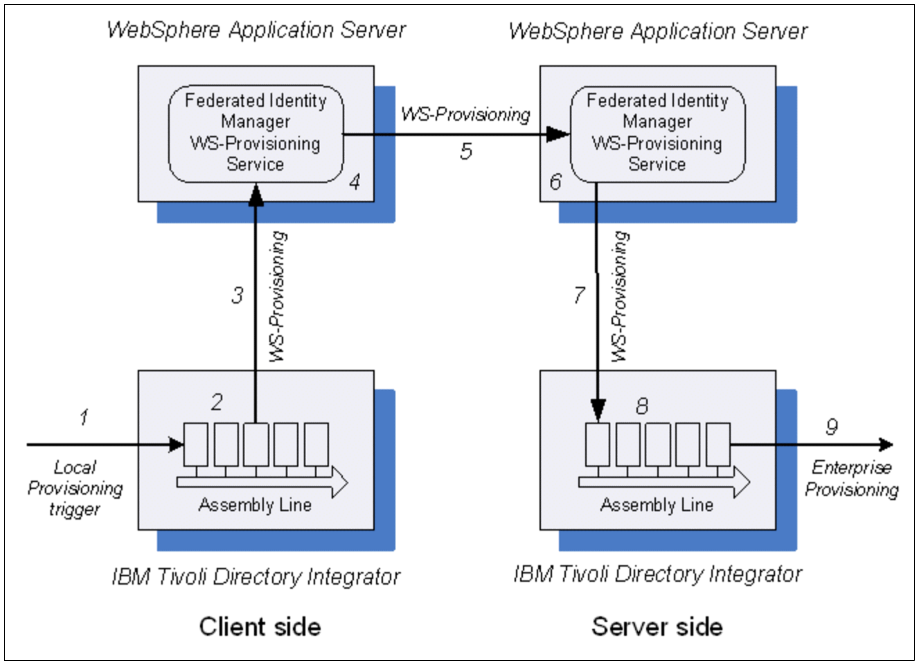

WebSphere Application Server
WebSphere Application Server
Federated Identity Manager WS-Provisioning Service
4
WS-Provisioning
5
Federated Identity Manager WS-Provisioning Service
6
3
WS-Provisioning
7
WS-Provisioning
1
Local Provisioning trigger
2
Assembly Line
8
Assembly Line
9
Enterprise Provisioning
IBM Tivoli Directory Integrator
IBM Tivoli Directory Integrator
Client side
Server side

*Figure 3-6 Tivoli Federated Identity Manager federated provisioning process flow*

In the figure:

1. An external event on the client triggers provisioning events. The event causes a Tivoli Directory Integrator AssemblyLine to be built and executed.
2. The AssemblyLine uses standard IBM Tivoli Directory Integrator connectors to collect the data necessary to form a WS-Provisioning message.

   Tivoli Directory Integrator provides a set of standard connectors that support the sending and receiving of Web services messages. Tivoli Federated Identity Manager configures these connectors by using the WS-Provisioning WSDL.
3. The Tivoli Directory Integrator connectors send a WS-Provisioning SOAP message to the Tivoli Federated Identity Manager WS-Provisioning service.
4. The Tivoli Federated Identity Manager WS-Provisioning service is a proxy server for WS-Provisioning messages that allows the addition of WS-Security policy for securing the Web services messages between partners.

   The addition of WS-Security policy to the WS-Provisioning service is optional, but typically is included in real-world deployments.

5. The WS-Provisioning service on the client side sends the message to the WS-Provisioning service on the server side.
6. The WS-Provisioning service on the server side receives the message and processes it.

   The processing includes handling of WS-Security information. The Tivoli Federated Identity Manager WS-Provisioning service can be configured to use either WebSphere Application Server WS-Security handling, or to use Tivoli Federated Identity Manager Web services security management (not shown here) to authorize the WS-Provisioning request.
7. The WS-Provisioning service sends the message to a configured connector on the local Tivoli Directory Integrator.
8. The Tivoli Directory Integrator connector receives the WS-Provisioning message and starts a configured assembly line.

   The AssemblyLine collects any local data that is required.
9. The AssemblyLine initiates local provisioning.

### WS-Provisioning service

The Federated Identity Manager WS-Provisioning provisioning service is a Java 2 Enterprise Edition (J2EE) application. The purpose of this application is to route WS-Provisioning messages to a destination provisioning service, and to act as a security proxy for incoming and outgoing WS-Provisioning SOAP messages. The provisioning service off-loads the cost of securing messages from the client and server WS-Provisioning endpoints that make use of the proxy. The provisioning service uses WS-Security and WS-Trust to secure WS-Provisioning messages. The WS-Provisioning specification does not require that messages be secured, but real-world deployments combine WS-Provisioning with WS-Security and WS-Trust to ensure the security and integrity of the messages, as follows:

- For outgoing messages, the application uses WebSphere Application Server to add WS-Security to the SOAP message and then proxies the secured message on to its destination. An example of a destination is another provisioning service.
- For incoming messages, the application uses either WebSphere or the Tivoli Federated Identity Manager Web Services Security Manager (WSSM) to authenticate the user by using the WS-Security information contained on the SOAP message. When the user has been authenticated, the application removes the WS-Security information and proxies (sends) the SOAP request on to its destination. The destination is a configured WS-Provisioning endpoint. An example of a destination is a Tivoli Directory Integrator assembly line.

### *Configuration of provisioning service*

Configuration of Tivoli Federated Identity Manager provisioning includes integration with Tivoli Directory Integrator and configuration of the provisioning service. Securing of the provisioning service is optional, but is typically included in the configuration steps:

- Deploying the IBM Tivoli Directory Integrator file
- Configuring the provisioning service (Details are in the following list of steps.)
- Securing the provisioning service

To configure the provisioning services, complete the following steps:

1. If you are configuring the provisioning service into a WebSphere Application Server cluster environment, you must deploy the provisioning service EAR file.

   > **Note:** If you are configuring the provisioning service into a WebSphere Application Server single-server environment, skip this step (continue with step 2.

   The Tivoli Federated Identity Manager provisioning service EAR file is installed as part of the Tivoli Federated Identity Manager runtime component. When you install the runtime component onto a WebSphere Application Server Network Deployment, as part of a cluster environment, the provisioning service file is also installed on the deployment manager. This file in installed as:

   ```
   provisioning/itfim-provisioning.ear
   ```

   When you deploy the runtime component, the Tivoli Federated Identity Manager runtime files are programmatically pushed from the deployment manager out to each node in the cluster. However, the provisioning service EAR file is not pushed to each node. You must manually copy the provisioning file to each of the target nodes.

   For each node:

   a. Create the directory `provisioning` under the Tivoli Federated Identity Manager runtime installation directory.

   b. Copy the `itfim-provisioning.ear` file from the deployment manager to the `provisioning` directory on the node.

2. Specify the proxy URL configuration.

   The proxy URL is configured using the custom properties feature of the Tivoli Federated Identity Manager runtime. Use the Tivoli Federated Identity Manager console to specify custom properties.

The default custom runtime properties file contains a placeholder entry for the proxy service URL. The default setting is:

```
provisioning.proxyDestinationURL =
http://your.provisioning.endpoint:9999/wsp/wspservice
```

On the client side, you must change this value to point to the partner WS-Provisioning service.

On the server side, you must change this value to point to the local Tivoli Directory Integrator service.

To change the value:

a. Log in to the Tivoli Federated Identity Manager administration console.

b. Select **Tivoli Federated Identity Manager** → **Domain Management** → **Runtime Node Management**.

   The Runtime Nodes portlet is displayed.

c. Click **Runtime Custom Properties**.

   The Runtime Custom Properties portlet is displayed.

d. Select the table row that contains the Name entry `provisioning.proxyDestinationURL`.

e. Replace the default entry in the Value column with the value of your provisioning endpoint:

   - On the client side:

     For example, when the host system of the provisioning service is `sp.example.com`, and WebSphere Application Server listens on port 9080, an example value is:

     ```
     http://sp.example.com:9080/wsp/wspservice
     ```

   - On the server side:

     For example, when the host system of the local IBM Tivoli Directory Integrator service is sp.example.com, and the IBM Tivoli Directory Integrator assembly line processes requests on port 8888, an example value is:

     ```
     http://sp.example.com:8888/wsp/wspservice
     ```

f. Click **OK**.

**Note:** The provisioning service does not require that a Tivoli Federated Identity Manager single sign-on federation be created and configured. However, in some cases, you might want to also configure a Tivoli Federated Identity Manager single sign-on federation. The provisioning service can be used in conjunction with F-SSO to enable federated user accounts to be provisioned.

## 3.4 Tivoli Directory Integrator

IBM Tivoli Directory Integrator manages the technicalities of connecting to and interacting with the various data sources that you want to integrate, abstracting away the details of their APIs, transports, protocols and formats. Instead of focusing on data, Tivoli Directory Integrator lifts the overall view to the information level, enabling concentration on the transformation, filtering and other business logic required to perform each exchange.

Tivoli Federated Identity Manager uses Tivoli Directory Integrator to support enterprise-specific provisioning systems. Tivoli Directory Integrator is a tool for integrating disparate data sources. A data source is accessed by using an Tivoli Directory Integrator *connector* that is specific to the type of data source. For example, an *LDAP connector* is used by Tivoli Directory Integrator to access LDAP directories.

Tivoli Directory Integrator supports functional components. A functional component is a unit of logic that is similar to a connector but without the built-in logic that is required to interface with a specific type of data source. Tivoli Directory Integrator connectors and functional components are configured into an *AssemblyLine* in order to receive data, manipulate the data, and transport the data. Connectors and functional components present a common interface to AssemblyLines so that data can be processed in a uniform manner.

AssemblyLines support the mapping of data between connectors and functional components. This means that the outputs from one connector become the inputs to another connector. Complex mapping, including the use of JavaScript, is also supported.

Tivoli Directory Integrator *event handlers* or *connectors*, when running in *server mode*, can be triggered by an external event. When triggered, the handlers or connectors start an AssemblyLine. For example, the Tivoli Directory Integrator LDAP event handler triggers a configured AssemblyLine when LDAP directory changes match a previously established selection criteria.

### Config Editor

The **`ibmditk`** command is used to start the Tivoli Directory Integrator *Config Editor*. The Config Editor is used to create various Tivoli Directory Integrator elements that might be required when using Tivoli Directory Integrator with Tivoli Federated Identity Manager provisioning service. For online documentation about the Config Editor, go to:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.IBMDI.doc/usersguide101.htm`

### Server

After the Tivoli Directory Integrator configuration has been finalized, deploying the solution is done with the Tivoli Directory Integrator Server script file, `ibmdisrv`. This script file sets up the Java VM environment and then launches the Server.

In addition to commands placed in the config file itself, you have a number of command-line options for controlling server behavior and its handling of the configuration. For `ibmdisrv` usage information go to:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.IBMDI.doc/usersguide177.htm`

## 3.5 Tivoli Access Manager for e-business

IBM Tivoli Access Manager for e-business is a product that provides policy management, access control enforcement, reverse-proxy protection of application servers, user registry, C and Java API libraries, and auditing capabilities. Tivoli Federated Identity Manager depends on the Tivoli Access Manager for e-business WebSEAL component for point-of-contact functionality for session management. Tivoli Federated Identity Manager is bundled with the full Tivoli Access Manager for e-business product including Tivoli Directory Server for user registry requirements.

**Note:** Before embarking on Tivoli Federated Identity Manager certification, Tivoli Access Manager for e-business certification is advisable. Tivoli Access Manager for e-business certification encompasses not only Tivoli Access Manager for e-business functionality and features, but also includes general concepts of authentication, authorization, certificate usage and management, policy management, and security architecture. The *Certification Study Guide: IBM Tivoli Access Manager for e-business 6.0*, SG24-7202 provides details about these concepts, which are also applicable for Tivoli Federated Identity Manager certification.

### 3.5.1 Policy Server

*Policy Server* is the component that provides policy administration and storage of policies within a master authorization database for a secure domain. Two types of administrative activities are handled by the Policy Server:

- Modifying the registry to define which objects participate in the secure domain
- Updating the authorization database with policy definitions

The Policy Server manages the master authorization database, which, in addition to resource policies, contains location information about other Tivoli Access Manager for e-business servers in the secure domain. Local replicas of the master authorization database are available for resource managers through a push/pull method initiated through the Access Manager runtime service. Each secure domain can have only one Policy Server.

### 3.5.2 IBM Global Security Kit

Tivoli Access Manager components communicate in a secure way over the network. Tivoli Access Manager provides data encryption through the use of the IBM Global Security Kit (GSKit) version 7.0.

The GSKit package also installs the iKeyman key management utility (gsk7ikm), which enables you to create key databases, public-private key pairs, and certificate requests. In other words, GSKit can be used to build a (somewhat trivial) PKI infrastructure. You must install GSKit before installing most other Tivoli Access Manager components. GSKit is a prerequisite to the Access Manager Runtime component, which is required on all Tivoli Access Manager systems with the exception of the Access Manager Attribute Retrieval Service, Access Manager for WebLogic Server, Access Manager Runtime for Java, or Access Manager Web Portal Manager.

The GSKit tool is often used to manage certificates that are used for WebSEAL's HTTPS communication. For performance improvement, WebSEAL supports SSL hardware acceleration. Utilizing the functionality of GSKit7, hardware acceleration can minimize the processor impact of SSL communications, improving the overall performance of the system.

### 3.5.3 WebSEAL

The Tivoli Access Manager for e-business WebSEAL component is a security manager for Web-based resources. WebSEAL is a high-performance, multi-threaded Web server that applies fine-grained security policy to the protected Web object space. WebSEAL can provide single sign-on solutions and incorporate back-end Web application server resources into its security policy. WebSEAL operates as a reverse-proxy.

### Supported WebSEAL authentication mechanisms

WebSEAL supports a large set of authentication methods. The most popular methods are:

- Forms-based authentication (user name and password)
- SPNEGO (Kerberos)
- Basic authentication (user name and password)
- External Authentication Interface (EAI)

**Note:** Tivoli Federation Identity Manager depends on the EAI authentication method for user authentication and user attribute data. Depending on federation configuration, an identity provider using WebSEAL as a point of contact accepts user data (that is user ID and user attributes) from WebSEAL. In a service provider federation configuration, Tivoli Federated Identity Manager sends user credentials, which contain user ID and attributes to the WebSEAL point of contact.

Further details about WebSEAL are available in the *Certification Study Guide: IBM Tivoli Access Manager for e-business 6.0*, SG24-7202 and Tivoli Access Manager for e-business online documentation available at:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.itame.doc_6.0/welcome.htm`

## 3.5.4 Tivoli Directory Server

IBM Tivoli Directory Server is an LDAP user registry that is included in the Tivoli Federated Identity Manager. Using Tivoli Directory Server as a user registry allows user attribute storage and retrieval when attribute management is required for a Tivoli Federated Identity Manager setup. Tivoli Federated Identity Manager also depends on a user registry for an alias service of SAML 2.0 and Liberty protocols. For details about Tivoli Directory Server setup and configuration online documents are available at the IBM Tivoli Directory Server information center:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.IBMDS.doc_6.0/welcome.htm`

## 3.5.5 Common Auditing and Reporting Service

Tivoli Federated Identity Manager can be configured to send audit data to an audit file or a Common Auditing and Reporting Service (CARS). The CARS subsystem transports and stores audit events to a common audit database that can be used to support operational reports from Crystal Enterprise or any other reporting tool. CARS also provides the capability to stage audit data to customizable report tables.

### Installing the CARS event server

At a high level, the steps to install the event server are:

1. Install the prerequisite products:
   - IBM DB2 Server
   - WebSphere Application Server
2. Review the preinstallation checklist for UNIX, Linux, or Windows operating system that includes verification of valid user and group permissions, and then validate the DB2 and WebSphere Application Server environment.
3. Determine the installation options.
4. Install the event server using either the interactive or silent installation.

   `<CARS install root>/serverInstall.log` file captures errors that might occur during installation.

### CARS configuration

To enable auditing, you must first configure the Tivoli Federated Identity Manager auditing settings.

Tivoli Federated Identity Manage auditing service is configured by using the console. You must have Tivoli Federated Identity Manager and the Common Auditing and Reporting Service (CARS) installed and configured correctly.

The CARS server root signer certificate must be imported into the Tivoli Federated Identity Manager keystore.

Under Domain Management within the Tivoli Federated Identity Manager console, select the domain you want to enable for the auditing service, and then:

1. In the Domain Management navigation area click **Auditing** to display the Audit Settings panel.
2. Select the Enable audit check box. You can select or deselect this check box at any time depending on whether you want to enable or disable auditing. The default setting is unchecked, that is, auditing is disabled.

3. Select the Tivoli Common Audit and Report Server radio button to send the audit records to the Common Audit and Report Service server.
4. Type the address for the Common Audit and Report Server in the Web Service URL field. For example:

   ```
   http://localhost:9080/CommonAuditService/services/Emitter
   ```

   The server can be either a secure server (HTTPS) or an unsecure server (HTTP).
5. Specify the location of the disk cache files in the Disk cache location field. The disk cache location specified here serves as a path prefix for where disk cache files are created. The Tivoli Federated Identity Manager auditing service generates an absolute path for runtime and management audit events as follows:
   - For runtime audit events:

     ```
     <Your specified path prefix>/<domain name>/<cell name>/<node
     name>/<server name>
     ```

   - For management audit events:

     ```
     <Your specified path prefix>/mgmt/<domain name>/<cell name>/<node
     name>/<server name>
     ```

   Type the appropriate information for your audit file settings in the Disk cache location field. Note the following information:
   - You can accept the default audit_location. This resolves to `<WebSphere profile path>/logs/fimaudit`.
   - Type a relative path which resolves to `<WebSphere profile path>/<your relative path>`.
   - Type an absolute path which is used as entered for the audit log file path.
6. Click **Web Service Security Settings**
   - If you are using HTTP, go to step 10 on page 155 to select the appropriate authentication.
   - If you are using HTTPS, continue with the next step to select your SSL settings.

7. From the Keystore drop-down menu select **DefaultKeyStore**.

   **Note:** You must import the Common Auditing and Reporting Service server root signer certificate to this keystore before performing this step. The Tivoli Common Auditing and Reporting Service supports only non-trusted keystores for security. This means that you cannot select a trusted keystore, such as DefaultTrustedKeyStore, from the list of keystores. Use a non-trusted keystore, such as DefaultKeyStore.

8. Type the password in the Keystore Password field.
9. Click **List Keys** and select the key you want to use.
10. Select the type of authentication, as listed in Table 3-15.

*Table 3-15 Type of authentication*

| Click | Required Actions |
|---|---|
| - | No action required. This is the default setting, not to use authentication. |
| BA | Type the user name and password in the Basic Authentication Username and Basic Authentication Password fields. |

11. Click any of the following buttons:
    - Click **OK** to save your changes and exit
    - Click **Apply** to save your changes without exiting.
    - Click **Cancel** to exit without saving your changes.

You must stop and restart the WebSphere Application Server for your changes to take effect.

## Securing CARS

If securing Common Auditing and Reporting Service (CARS) is a requirement, certain additional steps are required:

- Configuring the server
- Event server declarative security roles
- Securing the XML event store

Details about securing CARS can be found at:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tivoli.fim.doc_6.1/tfim61_audit73.htm`

### *Mapping security roles to users and groups*

The Common Auditing and Reporting Service Web service and the EventServer (Common Event Infrastructure) both define security roles. To send events to the Common Auditing and Reporting Service event server, these roles must map to a user or group that is defined in the user registry that is configured:

- Mapping Web service roles

  The Web service defines only one role named EventSource. This role defines the source of events to be submitted to the Common Event Infrastructure.

  The Web service role, EventSource, can be mapped to a user or group using the WebSphere Administration Console:

  a. Select **Applications** → **Enterprise Applications** → **CommonAuditService** → **Map security roles to users/groups**.

  b. Select **EventSource** role.

  c. Click one of the following items:

     - Look up users to map users
     - Look up groups to map groups

  d. Click **Search** to display the available users or groups list.

  e. Select the users or groups from the list and click the double-angle bracket (**>>**) to move the users or groups to the selected list. If the user registry is Local Operating System, then select root, for example.

  f. Click **OK** to add the selected users or groups to the Mapped users or Mapped groups list.

  g. Click **OK** and then save the changes. If you are in a WebSphere Application Server Network Deployment environment, be sure to select Synchronize changes with Nodes before saving the changes.

- Mapping Common Event Infrastructure roles

  The Common Event Infrastructure EventServer defines the following roles: eventAdministrator, eventConsumer, eventUpdater, eventCreator, catalogAdministrator and catalogReader. See event server declarative security roles for a description of each role, at:

  `http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.tivoli.fim.doc_6.1/tfim61_audit76.htm`

  Only the eventCreator role has to be mapped for the Common Auditing and Reporting Service C Client.

To map the eventCreator role to a user or group, use the WebSphere Application Server administrative console:

a. Select **Applications** → **Enterprise Applications** → **EventServer** → **Map security roles to users/groups**
b. Select **eventCreator**.
c. Click on one of the following items:
   - Look up users to map users
   - Look up groups to map groups
d. Click **Search** to display the available users or groups list.
e. Select the users or groups from the list and click the double-angle bracket (**>>**) to move the users or groups to the selected list. If the user registry is Local Operating System, then select root, for example.
f. Click **OK** to add the selected users or groups to the Mapped users or Mapped groups list.
g. Click **OK** and then save the changes. If you are in a WebSphere Application Server Network Deployment environment, be sure to select Synchronize changes with Nodes before saving the changes.

## 3.6 Conclusion

Depending on requirements, installing and configuring a Tivoli Federated Identity Manager can be complex. However, after the role of each component is understood, a plan can be quickly made for any particular setup. The major component for Tivoli Federated Identity Manager is WebSphere Application Server. For continuous availability, WebSphere Application Server Network Deployment is the preferred choice. Tivoli Federated Identity Manager also takes advantage of clustering to provide failover and load balancing capabilities.

Tivoli Federated Identity Manger supports the configuration of all major F-SSO protocols, including:

- SAML 1.x
- SAML 2.0
- WS-Federation
- Liberty

The Web services security management component offers a method for existing Web services applications to support various tokens for both consumption and generation.

The following tokens are supported by WSSM:

- SAML 1.x
- SAML 2.0
- Username
- Passticket
- X509
- Kerberos

Tivoli Federated Identity Manager requires a point-of-contact server. Therefore, Tivoli Access Manager for e-business is bundled with Tivoli Federated Identity Manager. In particular, the WebSEAL component is supported as a point of contact for Tivoli Federated Identity Manger. In addition, Tivoli Directory Integrator is included and can be used for provisioning services, which support the WS-Provisioning specification. Finally, for auditing requirements, Tivoli Federated Identity Manager provides the Common Auditing and Reporting Service component, which allows capturing of audit data within Tivoli Federated Identity Manager.

In the next chapter we discuss validating and testing of a Tivoli Federated Identity Manager setup.

4

# Test and validation

In this chapter, we discuss test and verification of a federated identity management solution between trusted business partners. We also talk about aspects of understanding how the user life cycle management of identities and the provisioning of user information is performed in a federation context. To help you better prepare for the test and validation we repeat important concepts and principles.

## 4.1 Federated single sign-on

A *single sign-on federation* is a *trust relationship* between two or more business partners. The trust relationship is built based on agreements that have been reached between the business partners. The agreements include both the definition of specific business policies and the adoption of agreed-upon technical standards. When the agreements have been defined, the administrator builds a trust infrastructure to support them. The building of a trust infrastructure requires a definition of a number of technical concepts.

- Identity provider and service provider roles
- Point-of-contact server
- Federation protocol profiles

In a federated single sign-on (F-SSO) solution a Web-based user authenticates to a federation business partner, *identity provider* (IdP), and has the IdP assert a relevant identity (and attributes) to any or all required *service providers* (SP) as part of the user's online federation experience.

Global sign-on itself is provided by a federated single-sign-on protocol that uses standard, interoperable means for multiple federation business partners to negotiate the presentation of credentials about a user from an identity provider to a (trusted) federation service provider. We have to understand and test for two distinct aspects:

- The first aspect is concerned with the format and content of the security token that is passed between the partners. The security token generated by the sending partner must be understandable by the receiving partner. Also, there must be an agreement as to what information is sent in the token and how it is interpreted. Typically, the security token format is bound to the single sign-on protocol (SAML protocols use SAML assertions, Liberty ID-FF protocols use Liberty specializations of SAML assertions). With Tivoli Federated Identity Manager, the security token generation and consumption is handled by the trust service while it is invoked internally by the single sign-on protocol service.
- The second aspect covers the single sign-on protocol, defining how the parties will communicate. A single sign-on server must know how a client will request a security token and how the token should be packaged and returned. The server must also know how a client will present an incoming security token in order to initiate an authenticated session. In Tivoli Federated Identity Manager, all single sign-on protocol messages are handled by the single sign-on protocol service.

## 4.1.1 Identity provider and service provider roles

Each partner in a federation has a role. The role is either *identity provider* or *service provider*.

### Identity provider

An identity provider is a federation business partner that vouches for the identity of a user. The identity provider authenticates the user, and provides an authentication token to the service provider. The identity provider handles the management of user identities in order to free the service provider from this responsibility.

### Service provider

A service provider is a federation business partner that provides services to users. Typically, service providers do not authenticate users but instead request authentication decisions from an identity provider. Service providers rely on identity providers to assert the identity of a user, and to manage user identities for the federation. Service providers can maintain a local account for the user, which can be referenced by an identifier for the user.

## 4.1.2 Point-of-contact server

The *point-of-contact* server is a proxy or application that is the first entity to process a request for access. In a typical deployment, the point of contact sits at the edge of a protected network in front of a firewall, such as in a DMZ. The point of contact provides endpoints that are visible to external users, such as browsers on the Internet. The point of contact receives access requests, and serves as the first component capable of evaluating the authentication credentials of the user that is requesting access to the protected network. In addition, the point of contact must handle *Web session management* for user sessions.

To satisfy the functional requirements for a point-of-contact server, Tivoli Federated Identity Manager leverages the extensive authentication and authorization capabilities of Tivoli Access Manager for e-business, which provides two components that can function as point-of-contact servers:

- Tivoli Access Manager WebSEAL
- Tivoli Access Manager Plug-in for Web Servers

Either component can serve as the point-of-contact server. In addition to authentication and Web session management, both can provide an entry point into the Tivoli Access Manager Authorization Services.

## 4.1.3 Federation protocol profiles

Tivoli Federated Identity Manager supports a number of Web single sign-on protocols. You should be familiar with the standards documents for these protocols before implementing a single sign-on federation. The protocols specify standards for data exchange and message processing. You should understand what information you are required to provide to your business partners, and what information your partner must provide to you.

The *IBM Tivoli Federated Identity Manager Configuration Guide Version 6.1.1*, GC32-1668 lists the supported standards and the location of the respective specification documents.

Each standard supports a unique range of single sign-on profiles. The profiles extend beyond specifications for achieving F-SSO, and can include other functions such as single logout and federation termination.

### Browser artifact single sign-on profile

The *browser artifact single sign-on* profile uses a SOAP back channel to exchange an artifact during the establishment and use of a trusted session between an identity provider, a service provider, and a client (browser). Note the following information:

- SAML supports browser artifact by default.
- WS-Federation does not support browser artifact.
- You can optionally select browser artifact for Liberty federations.

  For Liberty, you can optionally select both *browser artifact* and *browser POST* profiles when configuring an identity provider. You can select only one profile when configuring a service provider.

  For Liberty, when you select browser artifact, you are also prompted to enter the name of an encryption key for the trusted session. You must select a key even if you choose to not require the signing of assertions for other Liberty message communications.

### Browser POST single sign-on profile

The *browser POST single sign-on* profile uses a self-posting form during the establishment and uses a trusted session between an identity provider, a service provider, and a client (browser). Note the following information:

- For SAML 1, you can enable browser POST by enabling the IBM PROTOCOL extension when configuring a SAML 1 federation on an identity provider.
- WS-Federation supports browser POST by default.

- You can optionally select browser POST for Liberty federations.

  For Liberty, you can optionally select both browser artifact and browser POST profiles when configuring an identity provider. You can select only one profile when configuring a service provider.

### Liberty-enabled client or proxy single sign-on profile

A Liberty-enabled client or proxy (LECP) has, or knows how to obtain, the information that is required to be able to connect to the identity provider that the user (principal) wants to use with the service provider. A Liberty-enabled proxy is an HTTP proxy such as a Wireless Application Protocol (WAP) gateway that emulates a Liberty-enabled client.

### Register Name Identifier

The Register Name Identifier (RNI) profile updates the identifier for a specific user or principal. Liberty requires identity providers and service providers to exchange an alias (also called an identifier) to each user account, instead of exchanging the user's real account name. This ability enables account linkage while hiding the user account name.

When this optional profile is selected, the administrator must select the communication bindings to use between providers. The supported bindings are:

- HTTP redirect

  Updates to name identifiers are accomplished serially through the redirects. HTTP redirect is the default binding for both identity and service providers.

- SOAP/HTTP

  Updates to name identifiers are accomplished by direct exchanges between providers over a SOAP connection.

When you select the RNI profile during federation creation, you must select one of the binding types.

### Federation Termination Notification

This profile terminates account linkage across the federation for a specified user. This profile is disabled by default.

When this optional profile is selected, the administrator must select the communication bindings to use between providers.

The supported bindings are:

- HTTP redirect

  Termination of account federation is accomplished serially through the redirects.
- SOAP/HTTP

  Termination of account federation is accomplished by direct exchanges between providers over a SOAP connection.

When you select the federation termination notification profile during federation creation, you must select one of the binding types.

## Single logout for Liberty

This profile terminates all login sessions within the federation for a specified user. This profile is disabled by default.

When this optional profile is selected, the administrator must select the communication bindings to use between providers. The supported bindings are:

- HTTP redirect

  Logout of user sessions is accomplished serially through the redirects. HTTP redirect is the default binding for both identity and service providers
- HTTP GET

  Identity providers can use Image Tags to cause the browser to use HTTP GET to communicate the logout requests to the service providers. Logout requests are processed concurrently rather than serially. If a logout request fails, any remaining logout requests are unaffected, and are sent to the appropriate service provider. By contrast, when logout requests are processed serially (HTTP redirect) a failed logout request cancels any remaining logout requests.

  **Note:** This option is specified only on identity providers. Service providers cannot set this option.
- SOAP/HTTP

  Logout of user sessions is accomplished by direct exchanges between providers over a SOAP connection.

When you select the single logout profile during federation creation, you must select one of the binding types.

### Identity Provider Introduction

Identity Provider Introduction is a Liberty profile that enables a service provider to discover which identity providers are used by a user (Principal). The introduction profile relies on a cookie that is written in a domain that is common between identity providers and service providers in an identity federation network.

This profile is configured only on an identity provider.

### SAML 2 profiles

The SAML 2 protocol supports additional profile types.

- Single logout

  The *single logout* profile is used to terminate all login sessions currently active for a specified user within the federation. A user who achieves single sign-on to a federation establishes sessions with more than one participant in the federation. The sessions are managed by a session authority, which in many cases is an identity provider. When the user wants to end sessions with all session participants, the session authority can use the single logout profile to globally terminate all active sessions.

- Enhanced client or proxy

  An *enhanced client or proxy* (ECP) is a system entity that communicates with an identity provider and service provider on behalf of a user (client). For example, a user might request a resource from a service provider. The service provider might not know which identity provider to access in order to authenticate the user. The service provider can contact the ECP, which knows how to locate and access the appropriate identity provider. The ECP supports SOAP and reverse SOAP (PAOS) bindings during the processing of authentication requests.

- Name Management Identifier

  The *Name Management Identifier* profile manages user identities that are exchanged between identity providers and service providers. The profile enables identity providers to notify service providers when there is a change to the content or format of an identity for a given user (principal). The profile enables service providers to specify unique aliases for the principal, and to send those aliases to the identity provider to be used instead of the principal name. The profile also enables either provider to notify the partner when the provider decides to no longer issue or accept messages that use the principal's identity.

- Identity Provider Discovery

  The *Identity Provider Discovery* profile is used by service providers to discover which identity provider is used by a user (principal) during Web browser single sign-on. Some deployments have more than one identity provider, and the service provider must be able to determine which identity provider a principal uses. The Identity Provider Discovery profile uses a cookie that is created in a domain that is common between identity providers and service providers in a given deployment. The cookie contains the list of identity providers, and is called the common domain cookie.

## 4.1.4 Federated single sign-on functions

F-SSO is the process by which a user authenticates to a federation business partner (an identity provider, IdP) and has the IdP assert a relevant identity (and attributes) to any or all required (and trusted business partner) service providers as part of the user's online federation experience. Global sign-on itself is provided by a federated single-sign-on protocol. These protocols provide standard, interoperable means for multiple federation business partners to negotiate the presentation of credentials about a user from an identity provider to a (trusted) federation service provider.

We want to discuss several of the most relevant functionality to F-SSO:

- Push and pull SSO
- Account linking
- Where are you from
- Session management and access privileges
- Logout
- Credentials clean up
- Global good-bye
- Account de-linking

### Push and pull SSO

The two ways of performing SSO are *push* and *pull*. Push is available in SAML 1.x (with custom coding in Liberty ID-FF), and WS-Federation. Pull SSO is available in SAML 1.x and 2.0, Liberty ID-FF, and WS-Federation.

*Push SSO* means that the SSO exchange is triggered by a request to the identity provider, which then pushes a security token (or an artifact that can be used to obtain the security token) to the service provider.

*Pull SSO* means that the SSO exchange is triggered by a request to the service provider, which then pull’s a security token (or an artifact that can be used to obtain the security token) from the identity provider.

### Account linking

When a user has multiple login accounts at various sites or companies, navigating between these Web sites can be a cumbersome activity, not to mention the poor user experience. The user has to remember multiple site identity account names and passwords to access services on these Web sites. Account linking provides a simple mechanism for the user to link these distinct identity accounts that they have with different Web sites as long as the various companies or Web sites agree to this concept. The purpose of account linking is to deliver a single sign-on user experience with these various providers who are part of this agreement.

At a technical level, account linking is the process by which an identity provider and service provider agree on a *common unique identifier* (CUID), and then each bind their internal, local user identity to this common unique identifier. This approach allows the identity provider and service provider to refer to the user by the user's CUID during single sign-on without disclosing information about their local internal representation of the user.

### Where are you from

Some service providers might have trust relationships with multiple identity providers. This means that a user can possibly initiate SSO from one of many IdPs. For the service provider, the process of determining which IdP to request SSO from is referred to as *Where are you from?* (WAYF). This is a process by which a user may specify a preference for a given IdP for SSO purposes. This information is maintained by the SP so that it can easily determine, without user interaction, which IdP to request SSO from for future requests.

### Session management and access privileges

After a user has performed a single sign-on to a service provider, the SP is responsible for managing the user's local session at the SP. This process includes authorization decisions on the user's requested actions and also session management itself, such as logout or session time-out.

The implication is that the service provider is able to manage some level of attributes or credentials for a user. These attributes are used to determine a user's local access privileges. Access privileges may be asserted by the identity provider in the form of asserted attributes about a user, such as group membership. This information can be used by the service provider as an indication of the types of actions considered allowable by the identity provider (or, actions that will be honored by the IdP on the user's behalf). The service provider is able to honor or disregard these attributes as required for its local behavior.

### Logout

In some federation scenarios, the notion of single logout (or global logout) is also required, allowing a user to invoke a logout of all sessions asserted by a given identity provider. Global logout can be requested by a user from either the IdP or an SP; the process of global logout is controlled by the identity provider. The IdP is responsible for maintaining a list of all SP's to which the user has been signed on in a given session. The IdP will then send a logout request to each of these SPs on behalf of the user.

Global logout does not imply that local logout goes away. A user might want to log out of a session at a service provider without destroying the session at the identity provider. This process requires that the user know and understand the nature and workings of the federation. The more likely alternative to a local logout at a service provider is to provide a shorter session lifetime or inactivity time-out than is used in a standard, directly authenticated session.

### Credentials clean up

Logout, whether global or local, often implies the destruction of a session at a service provider. This session is often maintained at the edge of a network and might be independent of sessions with back-end applications. Back-end application sessions may be used to maintain a state between request and responses of a multiple step transaction. Logout, at both the identity provider, and service provider should ensure that not only edge sessions, but back-end application sessions (and session tracking artifacts), are destroyed.

### Global good-bye

Global good-bye deals with de-provisioning of a user's access rights and entitlements within a federation scenario. Global good-bye is used when a relationship between an identity provider and a service provider is broken, all of the user's attributes (including transactional, profile and provider specific attributes) that are relevant to the destroyed relationship are also destroyed. Note that federation relationships can be terminated in several ways:

- A user might chose to terminate the binding of an identity provider to an SP
- An IdP and SP might chose to no longer do business together, breaking the binding for all of the IdP's users.

### Account de-linking

Account de-linking is the process by which the common unique identifier is destroyed, removing the ability of an IdP and SP to uniquely refer to a given user. One result of account de-linking is that a user will no longer experience SSO from the IdP to the SP. Note that account de-linking is independent of how a user's account or registry record was created at the service provider, meaning that account de-linking is possible whether an account was explicitly created by a user and then linked, or created based on provisioning from the IdP to the SP. After de-linking an account, a user or SP can choose to link an account with a different IdP, or the SP can choose to resume direct authentication of the user.

## 4.2 Web services security management

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

The Web services security management component provides functions for:

- Web service providers, which have to process inbound security tokens using a token consumer.
- Web service clients, which have to create outbound security tokens using a token generator.

The processing and creation of the tokens is performed 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.

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 deployment of a F-SSO environment. Unlike the other Tivoli Federated Identity Manager components, the Web services security management component does not require the use of the Tivoli Access Manager WebSEAL component.

In this section, we describe the Tivoli Federated Identity Manager Web services security management component and validation instructions required for creating a connection between a previously created and configured Web service application and the Web services security management component.

### 4.2.1 Web service application

The general steps involved in enabling a Web service application to be used with the Web services security management component are as follows:

1. Configure your Web service application to use the Web services security management component.
2. Configure the Tivoli Federated Identity Manager trust service to be used with your Web service application.

Prior to using the Web services security management component with your Web service application, you must perform the following steps:

1. Install the Web services security management component.
2. Perform configuration tasks for the component and its required software, including:
   a. Configuring WebSphere Application Server.
   b. Updating configuration batch and script files.
   c. Configuring Tivoli Access Manager, if you plan to use the authorization function of the Web services security management component.
   d. Configuring keystores and certificates, if keys or certificates will be used with the tokens that you plan to validate or exchange.

For more information about Web services, refer to *Federated Identity Management and Web Services Security with IBM Tivoli Security Solutions*, SG24-6394.

## 4.2.2 Web services security

Web services security (WS-Security) defines a standard set of SOAP extensions that can be used when building secure Web services to implement integrity and confidentiality. This allows for sending security tokens to authenticate requests and signing data to ensure data integrity and verify sender. To ensure privacy of data, the data is encrypted. WS-Security is used to accomplish end-to-end message content security.

For more information about the SOAP message security specification, refer to *Web Services Security: SOAP Message Security 1.0*, available at:

http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

This standard defines a set of SOAP extensions, seen in Figure 4-1 on page 171, that provide the ability to:

- Send security tokens as part of a message.
- Include an XML Digital Signature as part of a message.
- Encrypt all or part of the message using XML Encryption.

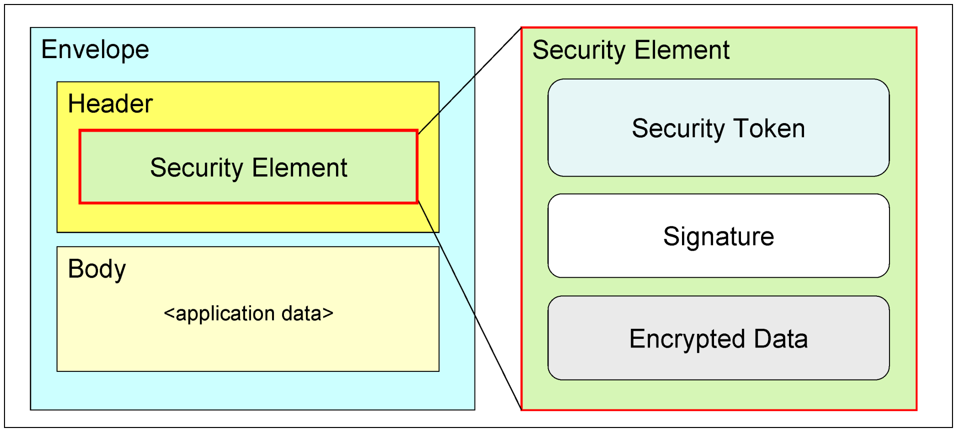


*Figure 4-1 SOAP message security, extensions to the header*

Web service requests are handled differently depending on whether you configure your environment to use the Web services security management component.

## Configuration without Web services security management

If the Web services security management component is not installed, or is installed but not enabled, the client application invokes the Web service application without any interaction with Tivoli Federated Identity Manager.

## Configuration with Web services security management

With the Web services security management component installed and enabled, additional security tokens are supported and Tivoli Federated Identity Manager can make an authorization decision on the Web services request before the underlying Web service is called.

In addition, the identity provided by the security token of the Web service application can be mapped to another identity, or the type of security token changed to one that is recognized by the application, before the Web service application is called.

The Web services security management functionality may be invoked at the Web service client, the Web service, or both.

## 4.2.3 Web service trust

When used with the Web services security management component, the Tivoli Federated Identity Manager trust service can manage the handling of security tokens that are part of the requests made to Web service applications.

The trust service processes requests based on the identity of the requestor (who is making the request) and the destination of the request (what is being requested). To convey this information, the request is structured according to the WS-Trust standard format that is defined for Security Token Requests. The request includes defined fields such as *Issuer* and *AppliesTo*:

- The *Issuer* is the issuer of the request. In the Web services security management component, the issuer is either the WSSM token generator or the WSSM token consumer.
- The *AppliesTo* field identifies the Web service that is the destination of the request.

The trust service uses the Issuer and the AppliesTo fields, along with other requests for specific WS-Trust functions, in order to determine how to process the request.

To process tokens, the trust service uses:

- Modules, which are plug-ins for handling different token format conventions
- Module instances, which are combinations of a module with optional parameters that provide instructions for specific processing
- Module chains, which are groups of module instances that are configured for use together

The general process used by the trust service is as follows:

1. Receive the RequestSecurityToken.
2. Determine which module chain should be used to process the request.
3. Invoke, in order, each module in the chain.
4. Use the resulting data to create a RequestSecurityTokenResponse.
5. Return the RequestSecurityTokenResponse.

## 4.2.4 Authorization services

When used within the context of Web services security management, the trust service can be configured with authorization services. The authorization services may be used to determine whether a user (as validated and identified by the trust service) is authorized to access requested resources. This step allows an implementation-independent decision on the access of a Web service; that is, whether the Web service exposes a J2EE-based resource, a CICS resource, or some other proprietary resource does not matter.

## 4.3 Federated identity provisioning

Federated identity provisioning extends provisioning management activities beyond an internal trust domain. Federated identity provisioning makes possible the extending of local account provisioning at an identity provider to include federated account provisioning out to multiple service provider partners. A service provider, when notified of the federated provisioning request, can perform the local provisioning necessary to supply its service to the specified requester.

In this section, we address a mechanism by which local provisioning systems are linked in a Tivoli Federated Identity Manager solution.

### 4.3.1 Overview

Provisioning is a key element in identity management for enterprises. When used with provisioning of account data and authentication credentials, provisioning solutions generally are of two types: *runtime* (or just-in-time) and *a priori* provisioning.

*Runtime provisioning* solutions are also referred to as enrollment solutions because a user is registered, or enrolled, for a set of services, as part of the fulfillment of a single sign-on request. Sometimes this is referred to as silent registration because the users do not see a separate registration/enrollment step in their user experience.

*A priori provisioning* is the process by which a user account creation request can be sent to federation business partners outside of the scope of a single sign-on request. This approach allows both the IdP and SP to create local accounts and registry records for a user in response to some action at the IdP. *A priori* provisioning is often triggered by an account creation event at the IdP. *A priori* provisioning can also be triggered by other events, such as a change in a user's status that in turn gives the user access to more business partner resources; or a subscription event by a user, signing up for services that the IdP in turn outsources to a third-party service provider. Note that as with runtime provisioning, a common user identifier is established for a user automatically as part of *a priori* provisioning.

Runtime, or just-in-time provisioning allows a service provider to create a user account or record in that user’s local registry in response to a single-sign-on request from a trusted IdP. This might happen when an SP receives an SSO request from a trusted IdP but does not have any record of the user claimed in the SSO request. Instead of rejecting the SSO request, the SP might choose to create a user record based on the claimed common unique identifier (CUID). The CUID-local identity mapping is therefore established at this time; in fact, the SP is not required to ever establish its own, non-CUID local identity for this user.

Provisioning solutions allow the identity provider to create or update a user's transactional attributes, such as entitlements to service providers, as required. These attributes are typically managed by the user's identity provider.

### 4.3.2 Provisioning services

Provisioning services interact with both local identity management systems (such as Tivoli Identity Manager) and local data stores (access through identity services). Provisioning services are leveraged to federate local identity management systems across federation business partners and to provide federated management of identity data, including transactional and profile attributes.

WS-Provisioning describes the APIs and schemas necessary to facilitate interoperability between provisioning systems and to allow software vendors to provide provisioning facilities in a consistent way. The specification addresses many of the problems faced by provisioning vendors in their use of existing protocols, commonly based on directory concepts, and confronts the challenges involved in provisioning Web services described using WSDL and XML Schema.

Note the following information:

- At the *communication layer*, SOAP messages are being handled by the application server, in this case WebSphere Application Server or WebSphere Web services Gateway. All real communication is through the Web services handlers in the application server.
- At the *protocol layer*, the WS-Security header in the SOAP request is handled by the Tivoli Federated Identity Manager trust handler. It must read the WS-Security headers sent by the third-party solution (incoming) or include headers for the third-party solution (outgoing).
- At the *trust layer*, security tokens are being exchanged between Tivoli Federated Identity Manager and the third-party solution. In Tivoli Federated Identity Manager, this layer is handled by the trust service. The trust service exchanges security tokens with the third-party solution in the WS-Security header of SOAP requests (as handled by the trust handler).

SOAP security is used to protect WS-Provisioning messages, and the provisioning service acts as a secured Web service, accessing the IBM Tivoli Director Integrator in the back-end.

## 4.3.3 Provisioning architecture

The only components specifically related to provisioning are the provisioning service itself and the *identity management service*, which represents the enterprise identity management service, in this case the IBM Tivoli Directory Integrator, but it could also be a bespoke identity provisioning capability. The Tivoli Federated Identity Manager Alias Service and LDAP registry are also needed if provisioning for Liberty single sign-on with account linkage is required.

The Tivoli Federated Identity Manager components are:

- The Tivoli Federated Identity Manager WS-Provisioning Web service that runs on WebSphere Application Server
- The Tivoli Federated Identity Manager WS-Provisioning connector that runs on IBM Tivoli Directory Integrator

Both components provide a full implementation of the three interfaces defined by the WS-Provisioning standard.

A provisioning event is sent from the identity provider to the service provider through this sequence:

1. Some type of *provisioning trigger* at the IdP initiates a Tivoli Directory Integrator AssemblyLine. Tivoli Directory Integrator provides several mechanisms to start an AssemblyLine. The creation of a new entry in an LDAP directory is detected by a monitoring agent, a DSMLv2 request from IBM Tivoli Identity Manager, or another enterprise provisioning service, and so on.
2. The Tivoli Directory Integrator AssemblyLine collects data to form a WS-Provisioning message. The AssemblyLine can use any of the standard Tivoli Directory Integrator facilities for this including the many standard Tivoli Directory Integrator connectors.
3. The Tivoli Federated Identity Manager WS-Provisioning connector sends a WS-Provisioning message to the Tivoli Federated Identity Manager WS-Provisioning service.
4. The Tivoli Federated Identity Manager WS-Provisioning service uses the Tivoli Federated Identity Manager Trust Server to create a SAML token for a configured identity and uses the WebSphere SOAP security support to forward the WS-Provisioning message to the target service provider.

5. The Tivoli Federated Identity Manager WS-Provisioning service on the SP receives the message and forwards it to a configured WS-Provisioning connector on a local Tivoli Directory Integrator. This Tivoli Federated Identity Manager WS-Provisioning service may be configured to use Tivoli Federated Identity Manager Web services security management for identity validation and request authorization with Tivoli Access Manager.
6. The Tivoli Federated Identity Manager WS-Provisioning Tivoli Directory Integrator connector receives the WS-Provisioning message and starts a configured Tivoli Directory Integrator AssemblyLine.
7. The Tivoli Directory Integrator AssemblyLine on the SP collects whatever local data is required and initiates local provisioning, using an enterprise provisioning system such IBM Tivoli Identity Manager if necessary.

**Note:** WS-Provisioning messages sent between Tivoli Directory Integrator and Tivoli Federated Identity Manager do not include SOAP security headers because they are assumed to be in a trusted environment. The WS-Provisioning messages from Tivoli Federated Identity Manager-to-Tivoli Federated Identity Manager do use the SOAP security support of WebSphere Application Server.

## 4.3.4 Provisioning configuration

Configuration of Tivoli Federated Identity Manager provisioning includes integration with Tivoli Directory Integrator and configuration of the provisioning service. Securing of the provisioning service is optional, but is typically included in the configuration steps.

### Deploying the IBM Tivoli Directory Integrator file

Tivoli Federated Identity Manager provides a WS-Provisioning JAR file for use by Tivoli Directory Integrator. The file must be copied from the default Tivoli Federated Identity Manager directory to the IBM Tivoli Directory Integrator installation directory.

The Tivoli Federated Identity Manager runtime component is installed on a system that also hosts WebSphere Application Server. IBM Tivoli Directory Integrator can be installed on that same system or on a remote system.

## Configuring the provisioning service

The provisioning service must be configured as follows:

1. If you are configuring the provisioning service into a WebSphere Application Server cluster environment, you must deploy the provisioning service EAR file.
2. Specify the proxy URL configuration. The proxy URL is configured using the custom properties feature of the Tivoli Federated Identity Manager runtime.

## Securing the provisioning service

WS-Security configuration is used to protect messages between the client side provisioning service and the server side provisioning service. You can use WebSphere Application Server to add WS-Security on both the client side and server side. Optionally, on the server side only, you can use the Tivoli Federated Identity Manager Web services security manager (WSSM) to add WS-Security.

To secure the provisioning service, you must modify the provisioning service Web module that is distributed as part of the Tivoli Federated Identity Manager runtime component application. This topic describes how to modify the module.

When the Web module has been modified, you must then redeploy the Tivoli Federated Identity Manager runtime application. The redeployment steps vary depending on whether you have a WebSphere cluster environment or a stand-alone WebSphere Application Server.

### *Summary of tasks*

The tasks are:

1. Use a development environment, such as IBM Rational Application Developer, to update the deployment descriptors in the Tivoli Federated Identity Manager runtime component:
   a. Import the runtime EAR file into a development environment, in preparation for modifying the file contents.
   b. Update the Web Services Security deployment descriptors for the provisioning service.
   c. Export the modified descriptors (as the development project) to a new Tivoli Federated Identity Manager EAR file, and copy it to the proper location
2. Use the Tivoli Federated Identity Manager administration console to deploy and configure the updated Tivoli Federated Identity Manager runtime component:
   a. Unconfigure and deploy any existing runtime component.
   b. Deploy and configure the updated runtime component.

## 4.3.5 Provisioning verification

After deployment and configuration, you should be able to validate the provisioning functionalities: add, modify, and delete. For information about deployment and configuration topics, see *IBM Tivoli Federated Identity Manager Installation Guide Version 6.1.1*, GC32-1667 and *IBM Tivoli Federated Identity Manager Configuration Guide Version 6.1.1*, GC32-1668.

### Verifying provisioning user create

To verify, use the following steps:

1. On the client side system, log in to the Tivoli Access Manager for e-business management console by issuing the command **`pdadmin`**.
2. Create a test user.

   The creation of the user also occurs on the LDAP server. This user creation triggers a change in the change log number. This change is detected by the Tivoli Directory Integrator event handler on the client's listening application.

   The client-side IBM Tivoli Directory Integrator contacts the server-side IBM Tivoli Directory Integrator with a request to create the same user. The server-side IBM Tivoli Directory Integrator example customer component uses the Tivoli Access Manager Java administration API to provision a Tivoli Access Manager user into the server-side LDAP.
3. Verify that the user was successfully provisioned, as follows:
   a. Log in to pdadmin on the server-side Tivoli Access Manager system.
   b. Use **`pdadmin`** to display information about the user you just created.

### Modifying a demonstration user

To modify, use the following steps

1. On the client side system, log in to pdadmin.
2. Set the testuser account (created in the previous step) to the valid state.
3. Verify that the same user on the server-side also gets modified.

### Verifying provisioning user delete

To verify, use the following steps

1. On the client-side system, log in to pdadmin.
2. Delete the user that you created.
3. Verify that the user was successfully deleted, as follows:
   a. Log in to pdadmin on the server-side Tivoli Access Manager system.
   b. Use **`pdadmin`** to display information about the user you just deleted.

# 5 Troubleshooting

The troubleshooting process, in general, requires that you isolate and identify a problem, then seek a resolution. In this chapter, we provide information about resources and tools that you can use when identifying and resolving issues that might be related to a Tivoli Federated Identity Manager deployment.

We discuss the following topics:

- Problem determination
- Web services security management
- Backing up and restoring a domain

## 5.1 Problem determination

Troubleshooting topics for Tivoli Federated Identity Manager are organized according to the following sequence:

1. Understand the symptom or specific product feature.
2. Follow the “Troubleshooting checklist for Tivoli Federated Identity Manager” for the appropriate feature or symptom.
3. Collect diagnostic data.
4. Analyze diagnostic data.

### 5.1.1 Understanding the symptom

The first step in the troubleshooting process is to learn more about the problem symptoms and about the affected product feature.

The following aspects can help you acquire the conceptual information necessary to effectively troubleshoot problems within Tivoli Federated Identity Manager:

- Troubleshooting: Describe the issue completely.
- Connectivity problems: Narrow the issue through a process of elimination.
- Tivoli Federated Identity Manager: Learn about the affected feature within the software.
- Fixes and updates: Check the list of recommended updates and reported fixes.
- Messages: Read the entire message text and the recovery actions.
- Performance problems and failures: Characterize the problem correctly in terms of what the symptoms are.

### 5.1.2 Following the troubleshooting checklist

The “Troubleshooting checklist for Tivoli Federated Identity Manager” is part of the *IBM Tivoli Federated Identity Manager Problem Determination Guide Version 6.1.1*, GC32-2288. The online version of it can be obtained from:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tivoli.fim.doc_6.1/tfim61_pdg27.htm

## 5.1.3 Collecting data

Sometimes you cannot solve a problem simply by troubleshooting the symptoms. In such cases, you have to collect more diagnostic data. Let us look more closely at the various log files.

### Message and trace logs

Tivoli Federated Identity Manager message and trace logs are managed and stored by WebSphere Application Server.

#### *Message logs*

Message logs are text files in which system operations are recorded. You have to examine the following message log files:

- Application server default message logs:
  - JVM™ Log

    Log file location on UNIX and Linux platforms:

    `<WAS_HOME>/profiles/profile_name/logs/server_name/SystemOut.log`

    Log file location on Windows platforms:

    `<WAS_HOME>\profiles\profile_name\ logs\server_name\SystemOut.log`
  - IBM Service Log

    Log file location on UNIX and Linux platforms:

    `<WAS_HOME>/profiles/profile_name/logs/activity.log`

    Log file location on Windows platforms:

    `<WAS_HOME>\profiles\profile_name\ logs\server_name\activity.log`
- Integrated Solutions Console default message logs:
  - JVM Log

    Log file location on UNIX and Linux platforms:

    `<WAS_HOME>/profiles/default/logs/ISC_Portal/SystemOut.log`

    Log file location on Windows platforms:

    `<WAS_HOME>\profiles\default\logs\ISC_Portal\SystemOut.log`
  - IBM Service Log

    Log file location on UNIX and Linux platforms:

    `<WAS_HOME>/profiles/default/logs/activity.log`

    Log file location on Windows platforms:

    `<WAS_HOME>\profiles\default\logs\activity.log`

### *Trace logs*

Trace logging, or tracing, provides IBM software support personnel with additional information related to the condition of the system at the time a problem occurred.

In contrast to message logs, trace logs capture transient information about the current operating environment when a component or application fails to operate as intended. You have to examine the following message log files.

- Application server default trace log:

  Log file location on UNIX and Linux platforms:

  `<WAS_HOME>/profiles/profile_name/logs/server_name/trace.log`

  Log file location on Windows platforms:

  `<WAS_HOME>\profiles\profile_name\logs\server_name\trace.log`

- Integrated Solutions Console default trace log:

  Log file location on UNIX and Linux platforms:

  `<WAS_HOME>/profiles/default/logs/ISC_Portal/trace.log`

  Log file location on Windows platforms:

  `<WAS_HOME>\profiles\default\logs\ISC_Portal\trace.log`

## Configuring log settings

Settings for message and trace logs can be configured using the WebSphere Application Server administrative console. Message logging is enabled by default. Trace logging should be enabled only at the direction of IBM support personnel.

Both message and trace logging can be controlled from the WebSphere Application Server administration console by selecting: **Troubleshooting** → **Logs and Trace**

### *Configuring the JVM log*

You can modify the file name, location, file format, file size, logging start and stop times, number of logs to keep, and severity level to be logged in the JVM Log.

The JVM log, also referred to as `SystemOut.log`, is a standard WebSphere Application Server log used for messages. For detailed information, refer to the JVM log topics in the WebSphere Application Server Online Information Center at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp`

To configure the log, follow these steps:

1. Start the WebSphere Application Server administrative console and log in, if necessary.
2. Select **Troubleshooting** → **Logs and Trace** to open the Logging and Tracing page.
3. Select the name of the server that you want to configure.
4. Click **JVM Logs**.
5. Select the Configuration tab.
6. Scroll through the panel to display the attributes for the stream to configure.
7. Change the appropriate configuration attributes and click **Apply**.
8. Save your configuration changes.

### *Configuring the IBM Service log*

The IBM Service log is enabled by default. You can change this setting or modify the names, location, file size, and severity level to be logged in the log.

The service log, also referred to as the `activity.log`, is a standard WebSphere Application Server log used for messages. For detailed information about the log, refer to the service log topics in the WebSphere Application Server Online Information Center.

To configure the log, follow these steps:

1. Start the WebSphere Application Server administrative console and log in, if necessary.
2. Select **Troubleshooting** → **Logs and Trace** to open the Logging and Tracing page.
3. Select the name of the server that you want to configure.
4. Click **IBM Service Logs**. Then select or clear the Enable check box to enable or disable logging. The service log is enabled by default.
5. Set the name for the service log. The default name is `activity.log`. If the name is changed, the run time requires write access to the new file, and the file must use the `.log` extension.
6. Set the maximum file size, which specifies the number of megabytes to which the file can grow. When the file reaches this size, it wraps, replacing the oldest data with the newest data.
7. Set the message filter level to the desired state.
8. Save the configuration.
9. Restart the server to apply the configuration changes.

## Enabling trace logging

You can enable trace logging at server startup or on a running server. Also, review the trace components.

### *Enabling at server startup*

To enable trace logging at server startup, follow these steps:

1. Start the WebSphere Application Server administrative console and log in, if necessary.
2. Select **Troubleshooting** → **Logs and Trace** to open the Logging and Tracing page.
3. Select **Server** → **Diagnostic Trace**.
4. Click **Configuration**.
5. Then, select or clear the Enable check box to enable or disable logging. The trace log is disabled by default.
6. Complete the configuration as instructed by IBM support personnel. For additional information about the configuration settings, refer to the troubleshooting and trace log topics in the WebSphere Application Server Online Information Center.
7. Save the changed configuration.
8. To enter a trace string in order to set the trace specification to the desired state, follow these steps:
    a. Select **Troubleshooting** → **Logs and Trace** to open the Logging and Tracing page.
    b. Select the server you want to configure.
    c. Click **Change Log Level Details**.
    d. If the **All Components** selection has been enabled, you might want to turn it off and then enable individual components.
9. Click a component or group name. Refer to "Trace components" on page 185 for a list of Tivoli Federated Identity Manager components. *If the selected server is not running, it will not be displayed in the list.*
10. Enter a trace string in the trace string box. For example, to specify tracing for only the trust server, enter: `*=info: com.tivoli.am.fim.trustserver.*=all`

    For more information about trace strings, refer to the WebSphere Application Server Online Information Center.
11. Click **OK**.
12. Click **Save** to save your changes. You might have to restart the WebSphere Application Server for the change to become effective.

### *Enabling on a running server*

To enable trace logging on a running server, follow these steps:

1. Start the WebSphere Application Server administrative console and log in, if necessary.
2. Select **Troubleshooting** → **Logs and Trace** to open the Logging and Tracing page.
3. Select **Server** → **Diagnostic Trace**.
4. Click the Runtime tab.
5. Select the “Save runtime changes to configuration as well” check box if you want to write your changes back to the server configuration.
6. Change the existing trace state by changing the trace specification to the desired state.
7. Configure the trace output if you want to change from the existing trace.
8. Click **Apply**.

For detailed information about the log, refer to the WebSphere Application Server Online Information Center.

### *Trace components*

Table 5-1 lists the various selectable trace components.

*Table 5-1 Trace component names and descriptions*

| Trace component | Component description |
|---|---|
| com.tivoli.am.fim | All Tivoli Federated Identity Manager components |
| com.tivoli.am.fim.audit | Audit |
| com.tivoli.am.fim.fedmgr2 | Single sign-on protocol service (SPS) |
| com.tivoli.am.fim.kess | Key service |
| com.tivoli.am.fim.liberty | Liberty single sign-on protocol |
| com.tivoli.am.fim.management | Console management |
| com.tivoli.am.fim.mgmt | Console management |
| com.tivoli.am.fim.saml | SAML single sign-on protocol |
| com.tivoli.am.fim.saml20 | SAML 2.0 single sign-on protocol |
| com.tivoli.am.fim.soap | SOAP connections |
| com.tivoli.am.fim.sps | Single sign-on protocol service framework |

| Trace component | Component description |
|---|---|
| com.tivoli.am.fim.trust | Trust service client |
| com.tivoli.am.fim.trustserver | Trust service |
| com.tivoli.am.fim.wsfederation | WS-Federation single sign-on protocol |
| com.tivoli.am.fim.wssm | Web services security management |

## Viewing logs

The format of the logs determines how they can be viewed. In this section, you are directed to various topics in the WebSphere Application Server Online Information Center for more information:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp

### *JVM logs*

To view the JVM logs, you can use the WebSphere Application Server administrative console, which supports viewing from a remote machine, or use a text editor on the machine where the log files are stored. Refer to the "Viewing JVM logs" instructions in the WebSphere Application Server Online Information Center for more information.

### *IBM service logs*

The service logs are written in binary format. To view the log, you can use tools that are part of WebSphere Application Server. Refer to the "Viewing the service log" instructions in the WebSphere Application Server Online Information Center.

### *Trace logs*

Trace data is generated as plain text in basic, advanced, or log analyzer format. On an application server, trace data can be directed to a file or an in-memory circular buffer. If the circular buffer is used, the data would have to be dumped to a file before it could be viewed.

On an application client or stand-alone process, trace data can be directed to a file or to the process console window.

Refer to the "Interpreting trace output" instructions in the WebSphere Application Server Online Information Center.

## 5.1.4 Analyzing data

After you collect data from multiple sources, determine how that data can help you to resolve your particular problem.

To analyze the data, follow these guidelines:

- Determine which data sources are most likely to contain information about the problem, and start your analysis there.

  For example, if the problem is related to installation, start your analysis with the installation log files (if any), rather than starting with the general product or operating system log files.

- Have a clear understanding of how the various pieces of data relate to each other.

  For example, if the data spans more than one system, keep your data well organized so that you know which pieces of data come from which sources.

- Confirm that each piece of diagnostic data is relevant to the timing of the problem by checking time stamps.

  Note that data from different sources can have different time stamp formats. Be sure to understand the sequence of the different elements in each time stamp format so that you can determine when the different events occurred.

**Tip:** The specific method of analysis is unique to each data source, but one tip that is applicable to most traces and log files is to start by identifying the point in the data where the problem occurs.

## 5.1.5 Verifying the installation

The IBM Integrated Solutions Console (ISC) is a WebSphere Portal application that is designed to provide a common GUI for administering both IBM software and custom applications. Tivoli Federated Identity Manager (FIM) uses the ISC to manage and configure Federated Identity Manager domains, federation business partners, Web services security partners, keystores, and the trust service.

Use a browser to access the ISC and verify the installation as follows:

1. If you have just installed the Integrated Solutions Console, verify that the WebSphere Application Server named ISC_Portal has started. This server instance is used by the Integrated Solutions Console.

   On UNIX and Linux, use the following command:

   ```
   /opt/IBM/ISC/AppServer/bin/serverStatus.sh -all -username iscadmin
   -password passw0rd
   ```

   On Windows, use the following command:

   ```
   C:\Progra~1\IBM\ISC\AppServer\bin\serverStatus.bat -all -username
   iscadmin -password passw0rd
   ```

2. Verify that the Integrated Solutions Console is active and accessible by a browser. Open a browser and access the ISC login page. For example, when the host name is `idp.example.com`, access the following URL:

   ```
   http://idp.example.com:8421/ibm/console
   ```

3. Log in as `iscadmin`.

   The Integrated Solutions Console starts, and displays the Tivoli Federated Identity Manager plug-in.

## 5.1.6 Verifying federated partner configuration

When you create a federation for an identity provider, a partner is automatically created.

After you create a federation for a relying party, you can choose one of several options for configuring a partner. If you choose not to add a standard partner, you may later create a partner for the federation, and you will have to provide several configuration values. The Tivoli Federated Identity Manager console provides a wizard to guide you through this process.

The values are:

- Identity Provider Company name

  This value is the contact information.

- Security Token Issuer

  This value is used to set the `protocolID` and endpoint URL in `etc/feds.xml` and the Issuer field in the STS chain mapping configuration.

- Maximum allowable clock skew between hosts (seconds)

  This value is the maximum allowable clock skew between the relying party host and the identity provider host.

- Validate signatures on Information Card tokens

  This check box specifies that incoming security tokens must be signed.

- Type of signature validation key
  - Public key from the KeyInfo in the signature of the Information Card token
  - Public key from a keystore.

- Keystore

  This value is Tivoli Federated Identity Manager keystore containing the key.

- Keystore password

  This value is the password that is required to access the specified keystore.

- Key to select

  The wizard presents a list of key aliases (names) stored in the keystore. You must select the key to use.

Verify that all options are configured as required.

### 5.1.7 Disabling logging to enhance performance

When using Tivoli Federated Identity Manager with Tivoli Access Manager, you can improve performance on a service provider by disabling logging for the Tivoli Access Manager Policy Server. So, when you have completed your troubleshooting exercise be sure to disable logging again.

To reduce usage of the central processor unit (CPU), follow these steps:

1. Back up the Policy Server directory. For example, on Linux or UNIX:

   ```
   /opt/IBM/WebSphere/AppServer/java/jre/PolicyDirector
   ```

2. Open the following file in a text editor:

   ```
   /opt/IBM/WebSphere/AppServer/java/jre/PolicyDirector/PDJLog.properties
   ```

3. Disable message logging by setting the following parameter to false:

   ```
   baseGroup.PDJMessageLogger.isLogging=false
   ```

## 5.2 Web services security management

In this section, we list the locations of log files and describe the logging options that are especially useful when attempting to debug a problem with Web services security management. We also describe the use of TCPMON as a tool for monitoring Web services requests for this scenario, including where in the configuration to specify endpoints for directing traffic through TCPMON.

### 5.2.1 Using the logs for Web services security management

When installing the Tivoli Federated Identity Manager Web services security management software, and configuring a WebSphere Application Server for use with Web services security management, you specify the following JVM argument for the application server (the path might be different depending upon your installation directory):

```
-Dcom.tivoli.am.fim.svc.config.location=<INSTALL_HOME>/etc/logcfg_wssm.xml
```

The `logcfg_wssm.xml` file contains logging and tracing settings that can be very useful for debugging Web services security management related issues. When initially deploying and testing an application, a good practice is to edit this file and change the property traceLevel to the value DEBUG_MAX.

The trace file appears in the following location:

```
<INSTALL_HOME>/logs/tivoli-common/FBT/wssm/logs/trace.log
```

### 5.2.2 Using the logs for the Secure Token Service

After you have deployed your Tivoli Federated Identity Manager runtime, containing the trust service that will be used by the Web services security management components, you can modify the trace level by using the Tivoli Federated Identity Manager console from **Service Settings → Logging Settings**.

## 5.2.3 Using the WebSphere logs

The most common log file to inspect in WebSphere Application Server is the server's `stdout` log file called `SystemOut.log`. By default, the WebSphere Application Server logs for a stand-alone server are in the following locations:

- In Linux or UNIX:

  `/opt/IBM/WebSphere/AppServer/profiles/profile/logs/servername`

- In Windows:

  ```
  C:\Program
  Files\IBM\WebSphere\AppServer\profiles\profile\logs\servername
  ```

This file generally provides high-level information about application startup and critical errors but can be configured for more detailed tracing: From the WebSphere Application Server administration console navigate to **Troubleshooting** → **Logs and Trace** → **server1** → **Change log level details**.

## 5.2.4 Using TCPMON

TCPMON is a simple java GUI utility that allows you to look at on-the-wire Web services messages by configuring it as a listener and forwarding messages to their intended destinations. A lot of information is available about TCPMON on the Web. The intent here is not to teach you about TCPMON, but rather to show you where you can use it in a typical deployment.

Four locations exist in our Web services scenario, which we introduced in 2.1.1, "Federation example" on page 28, in which use of TCPMON would be relevant to see where messages are flowing. These are listed in Table 5-2.

*Table 5-2 Using TCPMON in our Web services scenario*

| **Web services message flow** | **Configuring an endpoint for TCPMON** |
|---|---|
| The Web services request originating from the JSP Web services client on the way to the gateway. | Use the JSP client interface to get the original endpoint (so that you know the port to forward TCPMON to), and to set the endpoint to point to TCPMON. This is a standard part of the Stock Quote client program. |
| Call to trust service from outbound side of Web services gateway to exchange Access Manager credential for signed SAML assertion. | Modify the `trust.service.url` parameter in the Token Generator configuration of the Web services gateway outbound binding configuration. |

| Web services message flow | Configuring an endpoint for TCPMON |
|---|---|
| Call from gateway to RBStocks. This is a signed and encrypted message, so there is not a lot to see. | Modify the service URL in the WSDL file being read by the gateway, and re-import the WSDL into the gateway. |
| Call from the RBStocks application server to the trust service to exchange the signed SAML assertion for a mapped, unsigned SAML assertion. | Modify the trust.service.url parameter in the Token Consumer configuration of the Stock Quote application. |

## 5.3 Backing up and restoring a domain

The Tivoli Federated Identity Manager console allows you to export a domain configuration. This task exports all of the configuration information for a domain.

The Tivoli Federated Identity Manager console also allows you to import a domain configuration. This task imports the files that are necessary to migrate an existing domain configuration into a new environment, such as new hardware, or when moving from a test deployment to a production deployment. The import task does not import the entire configuration of a domain.

When you want to import the entire configuration of a domain, such as when restoring an existing configuration into the same environment, you must perform additional steps. These additional steps enable you to perform disaster recovery when your hardware and network environment is unchanged:

1. Install Tivoli Federated Identity Manager into the environment that is to be restored.
2. Stop the WebSphere Application Server.
3. Create a directory with a name that matches the name of the domain you want to create. Place the directory in:

   ```
   $WAS_HOME/profiles/your_profile_name/config/itfim
   ```
4. Unpack the JAR file that contains the exported configuration. Place it in the directory:

   ```
   $WAS_HOME/profiles/your_profile_name/config/itfim/your_domain_name
   ```
5. Restart the WebSphere Application Server.
6. Using the Tivoli Federated Identity Manager administration console, create a domain on the server with the same name as the directory created previously as `your_domain_name`.

   Note that no existing information is overwritten.

### 5.3.1 Exporting configuration

Use this task to export the Tivoli Federated Identity Manager configuration. For example, use this task to make a backup of configuration properties, or use it to migrate or copy your Tivoli Federated Identity Manager configuration to another system or to send it to IBM support personnel in case of a troubleshooting exercise.

**Note:** When you export your configuration, only your domain and federation settings (such as keys and partner settings) are exported. This function does not export your WebSphere Application Server configuration.

You must first install and configure Tivoli Federated Identity Manager before you can export a configuration.

When you export a Tivoli Federated Identity Manager configuration, the complete configuration is automatically gathered into a JAR file. The file serves as a Tivoli Federated Identity Manager archive file.

To export a configuration:

1. Log in to the console and select **Tivoli Federated Identity Manager** → **Domain Management** → **Import and Export Configuration**. The Import and Export panels are displayed.
2. Click **Export Configuration**.
3. When prompted, specify the location where the configuration JAR file is to be saved. Click **OK**. Note the location you specify, so that you can access this file later when you have to import the configuration.

### 5.3.2 Importing configuration

Use this task to import the Tivoli Federated Identity Manager configuration. This task is useful when you want to restore a configuration or create a duplicate of the configuration as part of implementing a data center model.

You must first install Tivoli Federated Identity Manager on the server where you will import the exported configuration and configure it to match the installation on your existing server.

To import a configuration:

1. Log in to the console and select **Tivoli Federated Identity Manager** → **Domain Management** → **Import and Export Configuration**. The Import and Export panels are displayed.
2. Select the **Domain Name** of the domain into which the configuration archive is to be imported.
3. In the **Configuration Archive** field, enter the fully qualified path name to the Tivoli Federated Identity Manager configuration archive.
4. Click **Import Configuration**. The configuration is automatically imported.

### 5.3.3 Federation configuration

For Liberty federations, you can export the federation configuration to a metadata file.

**Note:** This action is supported for Liberty 1.1, Liberty 1.2, SAML 1, and SAML 2 federations only.

To export your federation properties follow these steps:

1. Log in to the Integrated Solutions Console. Click **Tivoli Federated Identity Manager** → **Configure Federated Single Sign-on** → **Federations**. The Federations panel is displayed.
2. Select a federation from the table.
3. Click **Export**. The browser displays a message window that prompts you to save the file containing the exported data. Click **OK**. The browser download window prompts for a location to save the file.
4. Select a directory and file name and click **Save**.
5. Place the file in an your backup store location.

## 5.4 Conclusion

In this final chapter, we discussed executing an organized approach for problem determination, including understanding your system, collecting and analyzing data, and verifying several sets of configuration information. We then discussed how to use log file information in a Web services security management deployment and how to backup and restore your Tivoli Federated Identity Manager domain data.

# Sample questions

In this appendix, we provide sample questions for Test 000-891.

This appendix contains:

## Questions

The following questions can assist you in studying for the certification test:

1. You are configuring an IBM Tivoli Federated Identity Manager (ITFIM) Federated-Single Sign-On (F-SSO) environment for a company acting as a service provider. the company requests that Common Auditing and Reporting Services (CARS) be used to maintain all audit information. The corporate security policy requires that an audit record be generated for the creation, deletion, and modification of all federations and partners within ITFIM. Which event type within CARS generates an audit record for all of these activities?
   a. IBM_POLICY_AUDIT
   b. IBM_POLICY_ADMIN
   c. IBM_SECURITY_FEDERATION
   d. IBM_SECURITY_MGMT_POLICY
2. A Web Service application is deployed on a WebSphere Network Deployment V6 cluster. The application requires a JAAS Subject containing attributes, which uniquely identify the user. Multiple clients are accessing this application from different partners. The WebSphere Web Services Gateway (WSGW) is deployed at the network boundary to prevent unauthorized requests from getting to the Web Service. Which deployment scenario is correct?
   a. The WS-Security runtime is configured for the Web Service. The Web Services Security Management (WSSM) SAMLA STS module is deployed to the WSGW. Custom trust chains are set up to handle the different partners and token types.
   b. The WSSM Token Generator is configured at the WSGW. Custom trust chains are configured to handle the different partners and token types. The Web Services Security Management (WSSM) SAMLA Login module is deployed on the WebSphere Application Server (WAS) node hosting the Web Service application.
   c. The Web Services Security Management (WSSM) SAMLA Login Module is deployed to the WebSphere Application Server (WAS) node hosting the Web Service. The WS-Security runtime is configured on the WSGW for the Web Service application. The WSSM Tivoli Access Manager Authorization module is deployed to the WAS application server hosting the Web Service.
   d. A Shared Library is configured on the WSGW to include the WSSM libraries. The wsdl2tfim tool is run on the Web Service WSDL to generate the Tivoli Access Manager authorization namespace. The Web Services Security Management (WSSM) SAMLA Login module is configured in the

IBM Tivoli Federated Identity Manager Trust Service to extract the required attributes to uniquely identify the user.

3. Which path shows the hierarchy of a protected object space that represents a Web Service?
   a. /itfim-wssm/wssm-default/servicename/operation
   b. /itfim-wssm/wssm-default/container/servicename/porttype
   c. /itfim-wssm/wssm-default/servicename/porttype/operation
   d. /itfim-wssm/wssm-default/container/servicename/porttype/operation
4. Which WebSphere Application Server command is used to enable Service integration Bus Web Services?
   a. wsadmin
   b. ejbdeploy
   c. uddiDeploy
   d. wasprofile
5. Which two tasks are required to configure a Service Integration Bus for inbound services? (Choose two.)
   a. Create a bus
   b. Configure a messaging engine
   c. Create a destination in the bus
   d. Install a SOAP over HTTP listener
   e. Create the endpoint listener and connect it to the bus
6. If the customer does not want to use the EAI header, how would you control whether IBM Tivoli Federated Identity Manager will use the EAI header for single logout (SLO) when talking to WebSEAL?
   a. Edit the sps.xml file to use the pdadmin command-line interface.
   b. Edit the runtime custom properties to force Federated Identity Manager to use the pdadmin command-line interface.
   c. Under EAI stanza in the WebSEAL configuration file, comment out the entry for the entry EAI-logout.
   d. Under EAI stanza in the WebSEAL configuration file, change the value for the entry EAIlogout to false.

7. To write a plug-in that implements a custom alias service, what are the two required public interfaces that have to be implemented? (Choose two.)
   a. com.tivoli.am.fim.service.CustomService
   b. com.tivoli.am.fim.alias.service.AliasServiceClient
   c. com.tivoli.am.fim.identity.service.client.IdServiceClient
   d. com.tivoli.am.fim.alias.service.AliasServiceClientFactory
   e. com.tivoli.am.fim.identity.service.client.IdServiceClientFactory
8. FIM_WS_Provisioning assembly line is configured for WS-Provisioning using IBM Tivoli Federated Identity Management (ITFIM) and stored in the /opt/custom/FIM/IDI/FIM_AL.xml file. IBM Tivoli Directory Integrator is installed under /opt/IBM/IBMDirectoryIntegrator on Solaris™ platform.

   Which command runs the assembly line FIM_WS_Provisioning on a Solaris platform?
   a. /opt/IBM/IBMDirectoryIntegrator/ibmdisrv -c /opt/custom/FIM/IDI/FIM_AL.xml -r FIM_WS_Provsioning
   b. /opt/IBM/IBMDirectoryIntegrator/ibmditk -c /opt/custom/FIM/IDI/FIM_AL.xml -r FIM_WS_Provsioning
   c. /opt/IBM/IBMDirectoryIntegrator/bin/ibmdisrv -c /opt/custom/FIM/IDI/FIM_AL.xml -r FIM_WS_Provsioning
   d. /opt/IBM/IBMDirectoryIntegrator/bin/ibmditk -c /opt/custom/FIM/IDI/FIM_AL.xml -r FIM_WS_Provsioning
9. A SAML V1.1 Federation is configured on Identity Provider (IDP) with a base URL of https://www.mycompany.com/FIM

   The following error is received:

   ```
   FBTSML005E The current user making the request is not authenticated.
   ```

   What is a possible reason for this error?
   a. The user account password is invalid.
   b. Authentication of the user failed at IDP.
   c. The junction is not configured to pass user name.
   d. The user account in IBM Tivoli Access Manager for e-business is disabled.

## Answers

The correct answers to the sample questions in this appendix are:

1. d
2. b
3. d
4. a
5. a, e
6. b
7. c, e
8. a
9. c

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

## IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 202. Note that several documents referenced here might be available in softcopy only.

- *Federated Identity and Trust Management*, REDP-3678
- *Propagating Identity in SOA with Tivoli Federated Identity Manager*, REDP-4354
- *Federated Identity Management and Web Services Security with IBM Tivoli Security Solutions*, SG24-6394
- *Enterprise Security Architecture Using IBM Tivoli Security Solutions*, SG24-6014
- *Certification Study Guide: IBM Tivoli Access Manager for e-business 6.0*, SG24-7202

## Other publications

These publications are also relevant as further information sources:

- *IBM Tivoli Federated Identity Manager Installation Guide Version 6.1.1*, GC32-1667
- *IBM Tivoli Federated Identity Manager Configuration Guide Version 6.1.1*, GC32-1668
- *IBM Tivoli Federated Identity Manager Auditing Guide 6.1.1*, GC32-2287
- *IBM Tivoli Federated Identity Manager Problem Determination Guide Version 6.1.1*, GC32-2288

- *IBM Tivoli Federated Identity Manager Single Sign-on Guide Version 6.1.1*, GC32-0168
- *IBM Tivoli Federated Identity Manager Web Services Security Management Guide Version 6.1.1*, GC32-0169

## Online resources

These Web sites are also relevant as further information sources:

- IBM Tivoli Federated Identity Manager documentation, (only available online as HTML version, either on the Tivoli publications Web site or through your local installation)

  http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tivoli.fim.doc/welcome.htm
- IBM WebSphere Application Server Version 6.0 information center, (only available online as HTML version, either on the WebSphere publications Web site or through your local installation)

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp
- Refer to the Tivoli software education Web site to find the appropriate courses and education delivery vendor for each geography, available at:

  http://www.ibm.com/software/tivoli/education
- General training information is available at the following Web site:

  http://ibm.com/training
- You can also refer to the existing *Test Preparation Roadmap for Tivoli Federated Identity Manager* at the following Web site:

  http://www.ibm.com/certify/tests/map891.shtml

## How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this Web site:

**ibm.com**/redbooks

## Help from IBM

IBM Support and downloads

**ibm.com**/support

IBM Global Services

**ibm.com**/services

# Index

IBM Redbooks

Certification Study Guide Series: IBM Tivoli Federated Identity Manager 6.1

(0.2"spine)
0.17"<->0.473"
90<->249 pages

# Certification Study Guide Series: IBM Tivoli Federated Identity Manager 6.1

**Helps you achieve IBM Tivoli Federated Identity Manager certification**

**Explains the certification path and prerequisites**

**Includes sample test questions and answers**

This IBM Redbooks publication is a study guide for the "IBM Certified Deployment Professional - IBM Tivoli Federated Identity Manager V6.1" certification test, test number 000-891, and is meant for those who want to achieve IBM Certifications for this specific product.

The IBM Tivoli Federated Identity Manager Certification, offered through the Professional Certification Program from IBM, is designed to validate the skills required of technical professionals who work with the implementation of the IBM Tivoli Federated Identity Manager Version 6.1 product.

This book provides a combination of theory and practical experience needed for a general understanding of the subject matter. It also provides sample questions that will help in the evaluation of personal progress and provide familiarity with the types of questions that will be encountered in the exam.

This publication does not replace practical experience, and it is not designed to be a stand-alone guide for any subject. Instead, it is an effective tool that, when combined with education activities and experience, can be a very useful preparation guide for the exam.

SG24-7783-00 ISBN 0738433322